

ON THE CONTROL OF
ASYNCHRONOUS SEQUENTIAL MACHINES WITH RACES

By

THOMAS E MURPHY

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1996

UNIVERSITY OF FLORIDA LIBRARIES

To Mom and Dad

ACKNOWLEDGMENTS

I would like to thank all the members of my Ph.D. supervisory committee, especially Dr. Jacob Hammer, whose support and guidance made finishing this work possible. I would like to thank my parents, James and Catherine Murphy, for their love and support throughout my school years. I would also like to thank all my friends for their confidence that I would some day graduate.

TABLE OF CONTENTS

ACKNOWLEDGMENTS.....	iii
ABSTRACT	vi
CHAPTERS	
1 INTRODUCTION.....	1
2 TERMINOLOGY AND BACKGROUND.....	8
2.1 Functions and Morphisms.....	8
2.2 Machines.....	10
2.3 Races.....	18
3 CONTROLLERS.....	24
3.1 Controllers.....	24
4 REACHABILITY.....	28
4.1 Reachable and Stably Reachable States.....	28
4.2 Reachability Sets.....	30
4.3 The Matrix of Stable Transitions.....	33
4.4 The Skeleton Matrix.....	37
4.5 Reachability for Families of Machines.....	39
5 EXISTENCE OF CORRECTIVE CONTROLLERS.....	41
5.1 Operation of Corrective Controllers.....	41
5.2 Corrective Controller Existence for a Machine.....	55
5.3 Corrective Controller Existence for a Family of Machines.....	57
6 EXAMPLES.....	65
6.1 Example 1.....	65
6.2 Example 2.....	69
7 OTHER RESULTS.....	74
7.1 Multiple Races.....	74
7.2 Isomorphic Machines.....	76
7.3 Matching Smaller Machines.....	85
7.4 General Families of Machines.....	87
7.5 Strictly Causal Controllers.....	88
7.6 Other Work.....	89
LIST OF REFERENCES.....	91

BIOGRAPHICAL SKETCH	95
---------------------------	----

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

ON THE CONTROL OF ASYNCHRONOUS SEQUENTIAL MACHINES
WITH RACES

By

Thomas E Murphy

August, 1996

Chairman: Dr. Jacob Hammer

Major Department: Electrical and Computer Engineering

A problem in the design of asynchronous sequential machines is the presence of races. Races arise due to the unequal propagation delays of different gates or delays in the signal paths of combinational logic used in the asynchronous machine.

Methods of race-free design are available, but techniques for eliminating the adverse effects of races after implementation has not been investigated. When races occur in a machine, the accepted course of action is to build a new machine based on a race-free design. We propose to take a different, and possibly more economical route. We propose to develop techniques for the design of external controllers that control the race afflicted machine so as to eliminate the adverse effects of the races.

The controller is another asynchronous sequential machine connected as a feedback controller with the race afflicted machine. The

combined closed loop system is operated in fundamental mode so as to ensure no new races arise due to the addition of the controller.

The main advantage of this approach is that it circumvents the need to replace the entire machine. The construction of a controller is in many cases more practical and more economical.

CHAPTER 1 INTRODUCTION

In some applications, including computer software or hardware engineering, one may encounter asynchronous sequential machines that exhibit undesirable behavior. One important example of such undesirable behavior is the appearance of a *critical race*. A critical race is a flaw in the operation of an asynchronous sequential machine, which causes the machine to exhibit unpredictable behavior. Critical races can occur as a result of malfunctions, design flaws, or implementation flaws. When a critical race occurs, the accepted course of action in the classical practice of computer engineering is to build a new machine, based on a race free design. In the present work we propose to take a different, and possibly more economical, route. We propose to develop techniques for the design of external controllers that control the race-afflicted machine so as to render its response predictable, and eliminate the adverse effects of the race.

The main advantage of this approach is that it circumvents the need to replace the entire faulty machine. Instead, one only needs to build a controller that drives the faulty machine so as to eliminate the effects of the race. The construction of the controller is in many cases more economical than the replacement of the faulty machine; moreover, in some cases (e.g., for systems that operate in remote or inaccessible locations), the replacement of the faulty machine may not be a practical option.

To be somewhat more precise, consider an asynchronous sequential machine Σ that exhibits undesirable behavior. Following the general practice of Control Engineering, we shall attempt to correct the behavior of the machine Σ by connecting it to another sequential machine C that serves as a controller, as depicted below.

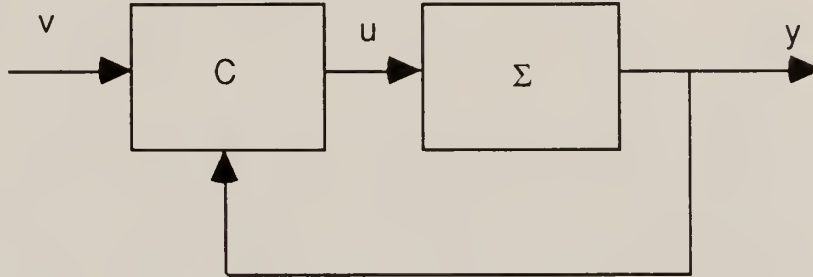


Figure 1.1. A controller C in a feedback connection with a machine Σ .

The purpose of the controller C is to drive the machine Σ so as to correct its behavior. In particular, C must eliminate the effects of races in Σ , and guarantee that the input/output response of the closed loop system is predictable. The controller C is sometimes called a *corrective controller*. We denote by Σ_c the input/output machine described by the closed loop system.

The main objective of the present work is to determine necessary and sufficient conditions for the existence of a controller C that counteracts the effects of critical races in a machine Σ . Whenever such a controller exists, we also provide techniques for its design.

An important question in this context is the question of when such a corrective controller can be found. In intuitive terms, the necessary and sufficient condition for the existence of a corrective controller is quite simple. First note that the objective of the corrective controller is to force the faulty machine to perform a prescribed set of desirable state transitions (in

a deterministic fashion). Let us consider, for simplicity, just one such desirable transition, say from state A to state B . Then, roughly speaking, a necessary and sufficient condition for the existence of a corrective controller is that, for every possible outcome of the critical race, there must be a succession of inputs that takes the faulty machine from state A to state B .

The controller functions as follows. Using its feedback arm, the controller detects the state that the faulty machine has reached as a result of the race. It then uses that information to create an input value that starts the machine along the path to the desired state B . At each state of the faulty machine along the way from A to B , the controller senses the state, and applies an input value that drives the machine one step "closer" to B . The overall effect of this process is to transform all states of the faulty machine along the path from A to B into unstable states, in such a way as to create, in effect, a direct transition from state A to state B . A slight further reflection along these lines also indicates that, in some cases, the order of the corrective controller will be substantially smaller than the order of the faulty system.

Consider a sequential machine Σ that has a critical race with q possible outcomes. In order to model the effects of the race, we represent Σ by a family $M = \{\Sigma_1, \dots, \Sigma_q\}$ of q sequential machine models, where each one of the models $\Sigma_1, \dots, \Sigma_q$ represents the behavior of the machine Σ for one possible outcome of the race. The controller C must control Σ so that the closed loop system Σ_c has the same response, irrespective of which one of the models $\Sigma_1, \dots, \Sigma_q$ is inserted into the loop for Σ . In this way, the closed loop system response is the same for all possible outcomes of the race, and hence the behavior of the closed loop system is predictable. In

addition to making the closed loop system predictable, we shall also use the controller C to assign other desirable characteristics to the closed loop system response.

In this context, it is important to emphasize the distinction between a *stable state* and an *unstable state* of an asynchronous sequential machine. In simple terms, a stable state is a state at which the machine may linger (if an appropriate input is provided); An unstable state is a state in which the machine can never linger, but can only pass in transient. In its path from one stable state to another, an asynchronous sequential machine may pass in rapid succession among several unstable states. Unstable states are inconspicuous to the user of the machine. The earlier statement regarding the fact that the closed loop system Σ_c has the same response for all members of the family M of models refers only to stable states. In fact, in crude terms, the controller C functions by transforming into unstable states the states at which the members of the family M differ from each other, and creating a set of stable states for the closed loop system that equalizes the response for all members of M .

In its treatment of families of sequential machines, the present work continues along the path charted by Hammer [14-18]. The main highlight of the present discussion is, of course, the treatment of critical races. This work develops controller design techniques, paying particular attention to the requirement that no new races be created when the controller C and the machine Σ are combined within the closed loop configuration. In more technical terms, the controller C is designed so that the controller and the system Σ always operate in fundamental mode.

This work is organized as follows. Terminology and some general background material are covered in Chapter 2. The existence of corrective

controllers and their structure are discussed in detail in Chapters 3 and 5 of this work. Chapter 4 is devoted to the introduction of the notion of reachability for sequential machines. The "matrix of transitions," a matrix of inputs, and the "skeleton matrix," a numerical matrix, both of which play an essential role in determining the existence and the structure of corrective controllers are also introduced in Chapter 4. Two comprehensive examples of the construction of corrective controllers are provided in Chapter 6. Future lines of research and some other results are discussed in Chapter 7. The presentation is for the case of a machine that is subject to a single critical race, but similar techniques can be used to treat machines with multiple races as well.

As discussed, the present work is concerned with the use of control techniques toward the abatement of the effects of critical races in existing systems. Such races may arise despite the availability of well established techniques for the design of race free asynchronous machines (e.g., Liu [27], Tracey [42], Kohavi [24], Maki and Tracey [29]), due to malfunctions of components, design flaws, or implementation flaws.

Various aspects of automata have been investigated by many people. Some of the early work on automata theory can be attributed to Turing [43]. Huffman [21][22] investigated various aspects of the synthesis of asynchronous relay circuits. Huffman also looked at race conditions and how to synthesize race-free circuits using state assignments. State assignment methods focus on the encoding of the state variables to avoid races. Huffman uses a method for assigning state variables so the resultant circuit is totally sequential. The state variables of adjacent states, the ones with direct state transitions, differ from each other by at most one variable. Unger [44] showed the existence of inherent hazards within fundamental

mode circuits and showed how to eliminate such hazards by inserting delays.

Other work on the synthesis of asynchronous machines was done by Mealy [33] and Moore [34]. They investigated state and circuit equivalence and provide methods for synthesizing asynchronous and synchronous sequential machines. Their models of sequential machines are two of the most commonly used today.

Single transition time state variable assignments, a method of race-free state assignment, was introduced in work done by Liu [27]. All internal transitions of a machine go directly from unstable states to terminal stable states. No sequencing through unstable states is permitted. The procedure uses assignments which correspond to an equidistant error correcting code. This is similar to mapping the rows of a flow table into the vertices of a unit n -dimensional cube, where the distances between every pair of vertices in the set is the same. The assignment is not guaranteed minimal, but works well with incompletely specified flow tables.

Tracey [42] extended Liu's work to assignment methods which maximize the operating speed of the circuit. The work done by Maki and Tracey [29] presents a shared row state assignment method. This method allows the sequencing through several unstable internal states before becoming stable after an input change. This method generally requires fewer state variables than single transition time assignments.

Other work regarding race-free synthesis using state assignment methods, and hazard-free synthesis can be found in McCluskey [31], Hazeltine [19], Armstrong, Friedman, and Menon [2], Hlavicka [20], Mago [30], Datta, Bandyopadhyay, and Choudhury [9], Chiang and Radhakrishnan [7], Piguet [38], Lavagno, Keutzer, and Sangiovanni-

Vincentelli [25], Oikonomou [35], Fisher and Wu [13], , and Chu, Mani, and Leung [8].

Yoeli and Rinon [47] and Eichelberger [10] proposed using three state logic models to analyze races in asynchronous circuits. Ternary, or three state logic, is helpful in analyzing certain aspects of races in machines. Ternary algebra provides an efficient method for the detection and analysis of hazards and races in sequential circuits. Ternary models were further worked on by Brzozowski and Yoeli [6] and Brzozowski and Seger [4][5].

Another method of avoiding races in asynchronous sequential machines involves creating a synchronization signal or clock pulse from the other signals in the machine. Two of these methods are given in Bredeson and Hulina [3] and Whitaker and Maki [45].

Other applications of control theory to the design of automata and asynchronous machines can be found in the literature on discrete event systems (e.g., Ramadge and Wonham [39-41], Wonham and Ramadge [46], Ozveren and Willsky [36], Lin and Wonham [26], Ozveren, Willsky, and Antsaklis [37], the references listed in these papers, and many others).

CHAPTER 2 TERMINOLOGY AND BACKGROUND

2.1 Functions and Morphisms

For two given sets X and Y , a function f with domain X and codomain (or range) Y assigns to each element $x \in X$ of the domain an element $f(x) \in Y$ of the codomain. We write this $f : X \rightarrow Y$. The *image* of $f : X \rightarrow Y$, $\text{Im } f$, is the set of all values $f(x)$, for $x \in X$. It is always a subset of the codomain. Next, we look at the effect of a function $f : X \rightarrow Y$ on subsets of X . The image of f under f of $U \subset X$ is defined to be the subset $f(U) = \{y \mid y \in Y \text{ and } y = f(u) \text{ for some } u \in U\}$. A function $f : X \rightarrow Y$ is said to be *surjective* when its image is its codomain.

Two functions f and g are equal, $f = g$, if they have the same domain, the same codomain, and the same value $f(x) = g(x)$ for each element x of their common domain.

For any set X , the *identity function* $1_X : X \rightarrow X$ is that function which maps each element $x \in X$ onto itself. If X is a subset of U , the *insertion* $i : X \rightarrow U$ is that function which assigns to each element of X the same element now in U .

The composition of two functions f and g , written fg , is the function obtained by applying the functions in succession, first g then f , provided the domain of f is the codomain of g .

A function $r : X \rightarrow Y$ is a *restriction* of a function $f : U \rightarrow V$ when $X \subset U$, $Y \subset V$, and $r(x) = f(x)$ for each $x \in X$. If r is a restriction of f then f is an *extension* of r .

A function $f : X \rightarrow Y$ is *injective* when f carries distinct elements of its domain to distinct elements of its codomain. A function $g : X \rightarrow Y$ is *surjective* when its image is the whole codomain Y . A function $h : X \rightarrow Y$ is *bijective* if it is both injective and surjective. In brief terms: Injective $X \rightarrow Y$ is a one to one map of X into Y , Surjective $X \rightarrow Y$ is a map of X onto Y , Bijective $X \rightarrow Y$ is a one to one map of X onto Y .

A function $X \times X \rightarrow X$ is called a *binary operation* on X . Let $*$ be a binary operation on a set X , and $'$ another such operation on a set X' . A *morphism* $f : (X, *) \rightarrow (X', ')$ is defined to be a function on X to X' which preserves the operation $*$ on X , i.e. $f(x * y) = f(x) * f(y)$ for all $x, y \in X$. The identity map of a set X is a morphism for any binary operation $*$ on X . If f and g are two morphisms, so is the composition fg . A morphism f is said to be a *monomorphism* if the function f is injective, an *epimorphism* if the function f is surjective, and an *isomorphism* if the function f is bijective.

Two sets, X and X' each with a binary operation $*$ and $'$ respectively, are called *isomorphic* when there exists some function f which is an isomorphism $f : (X, *) \rightarrow (X', ')$.

This terminology will be used systematically for each type of morphism, i.e. for morphisms of several operations, binary or otherwise. For more background in this area see MacLane and Birkhoff [28].

2.2 Machines

In the present subsection we introduce our notation and review some basic notions from the theory of asynchronous machines. An asynchronous machine is activated by sequences of characters, and generates sequences of other characters in response. We start therefore with a review of the basic terminology and notation related to sequences of characters.

An alphabet is a finite non-empty set A ; the elements of A are called *characters*. A *word* w over an alphabet A is a finite and ordered (possibly empty) string of characters from A . The empty word is usually denoted by the symbol λ . The set of all words over A is denoted by A^* , and we denote by A^+ the set of all non-empty words in A^* . The length $|w|$ of a word $w \in A^*$ is the number of characters in w . The concatenation of two words $w_1, w_2 \in A^*$ is a word $w := w_1w_2$ in A^* , obtained by appending the word w_2 to the end of the word w_1 . A non-empty word w_1 in A^* is a *prefix* of a word w_2 in A^* if $w_2 = w_1w$ for some $w \in A^*$; w_1 is a *suffix* of w_2 if $w_2 = w'w_1$ for some $w' \in A^*$. A non-empty word w_1 is a *subword* of a word w_2 if $w_2 = ww_1w'$ for some $w, w' \in A^*$, either w or w' could be empty. Finally, a *sentence* is a non-empty collection of words of A^+ , and we shall denote by $S(A)$ the set of all sentences of the alphabet A .

We now turn to a discussion of sequential machines. A sequential machine provides a mathematical model for a computing device with finite memory. A sequential machine has the following properties:

1. A finite set of inputs which can be applied to the device in a sequential order.

2. A finite set of internal configurations the device can be in. These are referred to as states, and in the physical device correspond to flip flop settings or combinations of bits in memory.

3. The present internal configuration and the input determine the next configuration the device achieves. If the device is deterministic, the next configuration is unique.

4. A finite set of outputs which are determined by the configuration of the device or the configuration of the device plus the input.

We can divide sequential machines into two main classes, asynchronous machines and synchronous machines. In an *asynchronous* sequential machine, the values of the output and internal variable may be read at any time, and the machine reacts to input changes instantaneously. A synchronous machine provides its output and internal variable values only at predetermined "clock" times, and its input variables are allowed to change only at these clock times. Our present work concentrates on asynchronous machines.

A *finite state machine* Σ is determined by a quintuple (A, Y, X, f, h) , where A , Y , and X are finite non-empty sets, and $f: X \times A \rightarrow X$ and $h: X \times A \rightarrow Y$ are partial functions. A point $(x, u) \in X \times A$ for which the partial function f is defined is called a *valid pair*. The valid pairs simply indicate which input values are permissible with a given state. In other words, the set of valid pairs is the domain of the function f . The set A is called the *input alphabet* of Σ , Y is the *output alphabet*; X is the *state set*, f is the *transition function*; and h is the *output function*. We assume that an initial state $x_0 \in X$ is provided with Σ .

A machine Σ is said to be complete if its transition function f and its output function h are functions rather than partial functions. A

deterministic finite state machine is often referred to as a Mealy machine. A machine is referred to as a Moore machine if its output function does not depend on the input variable.

A non deterministic finite state machine $\Sigma = (A, Y, X, f, h)$ consists of the following five objects:

- 1.) A finite, non-empty set A of allowable inputs called the *input alphabet*.
- 2.) A finite, non-empty set Y of allowable outputs called the *output alphabet*.
- 3.) A finite, non-empty set X of states. Any number of these states can be designated as initial or terminal states.
- 4.) A non deterministic transition function $f : \beta(X) \times A \rightarrow \beta(X)$, where $\beta(X)$ denotes the set of all subsets of X .
- 5.) A partial function $h : X \times A \rightarrow Y$, called the *output function*.

The state transition function f may accept sets of states as its values, rather than only single states, i.e., $f(x, a) = \{x_1, \dots, x_k\}$, where $x_1, \dots, x_k \in X$ and $a \in A$. Deterministic finite machines are a special subclass of non deterministic finite machines.

A machine Σ operates then as follows. It accepts an input sequence $u_0, u_1, u_2, \dots$ of elements of the input set A , and creates a sequence of states $x_0, x_1, x_2, \dots \in X$ and a sequence of output characters $y_0, y_1, y_2, \dots$ according to the recursion

$$(2.2.1) \quad \begin{aligned} x_{k+1} &= f(x_k, u_k), \\ y_k &= h(x_k), \quad k = 0, 1, 2, \dots \end{aligned}$$

Here, the integer k serves as the step counter. In an asynchronous machine, the step counter advances by one every time there occurs a state transition or a change in the input value. As usual, for consistency, we assume that the functions f and h have the same domain. Clearly, every

machine that follows the recursion (2.2.1) represents a causal system. When the output function h in (2.2.1) depends only on the state x_k (and does not depend on the input value u_k), then the machine Σ induces a strictly causal system.

Finally, we say that Σ is an *input/state machine* when $y_k = h(x_k)$ and h is a set isomorphism. In such case, the output at each step is isomorphic to the state of the machine at that step.

Using the transition function f , we can define a partial function $f^* : X \times A^+ \rightarrow X$ by setting $f^*(x, u) := f(\dots f(f(f(x, u_0), u_1), u_2) \dots, u_k)$ for a state $x \in X$ and a permissible input list $u = u_0 u_1 \dots u_k \in A^+$. Thus, the partial function f^* provides the last state in the path generated by the input string u . In the sequel, we shall somewhat abuse the notation and use the symbol f for f^* as well.

The following basic concept in the theory of asynchronous sequential machines plays an important role in our discussion. A valid pair $(x, u) \in X \times A$ of the machine Σ of (2.2.1) is a *stable combination* if $f(x, u) = x$. Thus, for a stable combination (x, u) , the state x does not change as long as the input character is u . A state x for which there is a stable combination is called a *potentially stable state*. States of Σ that are not potentially stable serve only as transition states, and the machine can never stop or linger in them. In idealized terms, the machine spends zero time in states that are not potentially stable. Such states are of no practical consequence, and can be ignored. The state sets of all the machines considered in the present work contain only potentially stable states.

An important restriction on an asynchronous sequential machine is the requirement that only a single variable affecting the operation of the machine be allowed to change its value at a time. A machine for which this

requirement is satisfied is said to operate in *fundamental mode*. For a machine to operate in fundamental mode, an input character can change only after the machine has reached a stable combination (so that state and input variables do not change values simultaneously); and, if the machine has several input variables, only one input variable may change at a time (e.g., Kohavi [24]). All machines and machine combinations discussed in this work operate in fundamental mode.

Consider the sequential machine Σ of (2.2.1). Let x be the state of the machine and let u be its input value. If (x,u) is not a stable combination, the machine Σ will go through a chain of transitions, which may or may not terminate. In fundamental mode, the input value u must be kept fixed while the chain of transitions is in progress. Clearly then, the chain of transitions terminates if and only if a stable combination (x',u) with the same input value u is encountered. When such a stable combination is encountered, we call x' the *next stable state* of x with the input value u . If a stable combination with the input value u is not encountered during the chain of transitions that passes through the pair (x,u) , then the machine Σ possesses an infinite cycle. Machines with infinite cycles are excluded from our present discussion.

As mentioned in the paragraph following (2.2.1), the step counter k advances one step during each state transition. Thus, if there are, for example, three transitions from (x,u) to the next stable combination (x',u) , then, although the input value is kept fixed at u , the machine is driven by the input list uuu in the course of the resulting transition chain. To simplify our notation, we shall disregard this fact in our ensuing discussion. We shall assume throughout that input values are kept constant during all transition chains (for fundamental mode operation),

and we shall count only the changes in the input value, rather than counting the steps of the machine.

The notion of next stable state leads to the notion of the *stable transition function*, which plays an important role in our discussion. Consider again the machine Σ of (2.2.1), and assume it has no infinite cycles. Then, every valid pair (x, u) of Σ has a next stable state x' . We can then define a partial function $s : X \times A \rightarrow X$ by setting $s(x, u) := x'$ for every valid pair (x, u) , where x' is the next stable state of x with the input value u . We call s the stable transition function of the machine Σ . Note that the stable transition function describes the operation of the machine Σ in fundamental mode, i.e., under the requirement that an input value be changed only after the machine has reached a stable combination. This guaranties that the state of the machine is fixed when the input value is being changed.

The operation of an asynchronous sequential machine Σ is determined by its stable transition function in the following way. The machine always starts from a stable combination (x_0, u_1) , at which the machine has been lingering for some time. Assume now that an input string $u := (u_0 u_1 \dots u_k) \in A^+$ is applied to the machine. This causes the machine to pass through a list $x_1, \dots, x_{k+1}$ of next stable states, where $x_{i+1} = s(x_i, u_i)$, so that (x_{i+1}, u_i) is a stable combination, $i = 0, 1, \dots, k$. The input list u is *permissible* if (x_i, u_i) is a valid combination for all $i = 0, \dots, k$. In fundamental mode operation, the input value is allowed to switch from u_i to u_{i+1} only after the stable combination (x_{i+1}, u_i) has been reached by the machine. This guaranties that the state of the machine during which the input value changes is well defined. The list of states $\{x_0, \dots, x_{k+1}\}$ is called the *path* induced by the input string u from the state x_0 . Note that the

intermediate states that the machine passes when moving from one stable combination to the next are not included in the path. The corresponding string of output characters generated by the input string u is $\{y_0, \dots, y_{k+1}\}$, where $y_i = h(x_i)$, $i = 0, \dots, k+1$. The fact that the input values up to step k determine the output values up to step $k+1$ is an indication of the strict causality of the machine.

Since the intermediate states the machine passes when going from one stable combination to the next are ignored in practice, the operation of the asynchronous sequential machine is best described by using the stable transition function s rather than the original transition function f of Σ . In formal terms, we use the stable transition function s to induce the sequential machine $\Sigma|_s$ that has the same input character set, the same output character set, the same state set, and the same output function as Σ , but its transition function is the stable transition function s of Σ . The machine $\Sigma|_s$ is called the *stable state machine* induced by Σ , and it is determined by the quintuple (A, Y, X, s, h) . The initial state of $\Sigma|_s$ is taken as the initial state of Σ .

Using the stable transition function s , we can define a partial function $s^* : X \times A^+ \rightarrow X$ by setting $s^*(x, u) := s(\dots s(s(s(x, u_0), u_1), u_2) \dots, u_k)$ for a state $x \in X$ and a permissible input list $u = u_0 u_1 \dots u_k \in A^+$, just as done for the case of the transition function f earlier. The partial function s^* provides the last state in the path generated by the input string u . We shall somewhat abuse the notation and use the symbol s for s^* below.

We can define a partial function $h^* : X \times A^+ \rightarrow Y$ in a similar manner by setting $h^*(x, u) := h(\dots s(s(s(x, u_0), u_1), u_2) \dots, u_k)$ for a state $x \in X$ and a permissible input list $u = u_0 u_1 \dots u_k \in A^+$, just as done for the case of

the stable transition function s earlier. The partial function h^* provides the last output generated by the input list u .

We can provide the entire output list by defining a function $h^+ : X \times A^+ \rightarrow Y^+$, where $h^+(x, u'u) := h(x, u')h^+(s(x, u'), u)$ for an input list $u \in A^+$, a character $u' \in A$, and a state $x \in X$. We could define the output functions h^* and h^+ using the transition function f instead of the stable transition function s , however, outputs are normally only valid in stable combinations of a machine, thus we use the stable transition function to define h^* and h^+ here.

It is necessary sometimes to compare two different sequential machines, we start by comparing states of a machine. Two states x and x' of Σ are distinguishable whenever there exists at least one finite input list which when applied to Σ , causes different output sequences depending on whether x or x' is the initial state. Two states x and x' of a machine Σ are *equivalent* whenever, for every possible input sequence, the same output sequence will be produced regardless of whether x or x' is the initial state.

Qualitatively, two states are equivalent if we cannot distinguish them by tests of input/output behavior on the machines concerned. If two states are equivalent, then so are the states they are taken into by an input string. If a machine Σ contains two equivalent distinct states, then one of these states is redundant, and can be omitted from the machine. A machine Σ is *minimal* or in *reduced form* if there are no distinct states in its state space which are equivalent to each other.

Now consider two machines $\Sigma = (A, Y, X, f, h)$ and $\Sigma' = (A', Y', X', f', h')$. The two machines Σ and Σ' are *equivalent* whenever, for every state in Σ , there is a corresponding equivalent state in Σ' , and for

every state in Σ' , there is a corresponding equivalent state in Σ . A state x of Σ *covers* a state x' of Σ' if every input sequence applicable to x' is also applicable to x , and the application of an input sequence to both Σ and Σ' when they are initially in x and x' respectively, results in identical output sequences. A machine Σ *covers* a machine Σ' whenever for every state x' of Σ' there is a corresponding state x of Σ such that x covers x' .

There are many good texts on various aspects of machines and automata theory, some are: Arbib [1], Eilenberg [11], Evans [12], Kalman, Falb, and Arbib [23], Kohavi [24], McCluskey [32].

2.3 Races

One problem with the design of asynchronous sequential machines is the presence of hazards and races. These problems arise due to the unequal propagation delays of different gates or delays in the signal paths of the combinational logic used in the asynchronous circuits. Static hazards are incorrect output changes in combinational logic that occur between changes of essential implicants. Critical races occur when two or more feedback signals (signals representing the internal memory) change together. Essential hazards are critical races between an input signal change and a feedback signal change. The present work concentrates on correcting the problems associated with critical races.

Our present work is within the context of computer engineering. In that context, sequential machines are implemented using digital circuits, which can only express an alphabet consisting of two characters, usually denoted 0 and 1. The implementation is achieved by the use of digital flip flops and logical gates. The set of states X then is represented by a family of

words made from the characters 0 and 1. The process of assigning these words to the states is the process of *state assignment*. The state assignment determines whether or not the resulting implementation will or will not have races. A race will result whenever the state assignment is such that a transition occurs between two states whose word representations differ by more than one character.

We turn now to a brief review of the notion of a race, one of the critical concepts in the theory of implementation for asynchronous sequential machines (e.g., Kohavi [24]). Recall that a race is a situation in an asynchronous sequential machine when two or more variables must change their value simultaneously. In practice, stray delays and the non-ideal characteristics of electronic devices make it impossible to achieve such a simultaneous change, and the variables will change sequentially. Moreover, it is usually impossible to predict the order in which the variables will change sequentially. A race is a *critical race* if the next stable state (or the next stable output value) of the machine depends upon the order in which the internal variables change. For a critical race, there are then several possibilities next stable states, Otherwise, the race is *non-critical*. A non-critical race does not affect the operation of the machine. Since the order in which the variables change their values is unpredictable, a critical race makes the response of the machine unpredictable as the machine passes through the race. A machine with a critical race will then have an unpredictable response for certain state-input combinations. An state-input pair (x,u) for which the next state of the machine is unpredictable is called a *critical race pair*, or, briefly, a *critical race*. Asynchronous sequential machines are always designed with the objective of obtaining a machine with no critical races. Nevertheless, a critical race

may occur in a machine as a result of a malfunction, an implementation fault, or a design fault.

Consider then a machine with a single critical race pair (x, u) . When the machine is at the state x and receives the input value u , the next state of the machine can be one of several options. Let $\{x_1, \dots, x_q\}$ be the set of all possible next states the machine may assume as a result of the pair (x, u) . The states $x_1, \dots, x_q$ are called the *outcomes* of the race. We then build a family $M = \{\Sigma_1, \dots, \Sigma_q\}$ of q sequential machines, all having the same input set, the same output set, the same state set, and the same output function; and whose transition functions are identical at all points except for the critical race pair (x, u) . At this point the value of the transition function f_i of the machine Σ_i is defined by $f_i(x, u) := x_i$, $i = 1, \dots, q$. In this way, each one of the members of the family M represents the behavior of our underlying sequential machine for one possible outcome of the race. Each individual member of M has, of course, no critical races. We shall refer to the family M as a *critical race family* of sequential machines. All the discussion of the present paper evolves around the notion of a critical race family.

Of course, a sequential machine may have more than one critical race, and a family of sequential machines can be built to represent such multiple critical races. The present work, however, is devoted to the study of critical race families that model the effects of a single critical race.

Next we discuss the effect of a critical race on the stable state machine $\Sigma|_s$ induced by a machine Σ .

(2.3.1) PROPOSITION. Let $\Sigma = (A, Y, X, f, h)$ be a machine, and let the pair $(x, u) \in X \times A$ be a critical race in Σ . The critical race (x, u) has no effect on the stable combinations of Σ .

PROOF. Let $z_1, z_2, \dots, z_n \in \{0, 1\}$ represent the internal variables of the states of the machine Σ . Recall a stable combination (x', u') is a pair for which the transition function is defined by $f(x', u') = x'$. In more specific terms, a stable combination is a state/input combination $(z_1 z_2 \dots z_n, u')$ where the transition function defined at the combination $(z_1 z_2 \dots z_n, u')$, $f(z_1 z_2 \dots z_n, u') = z_1 z_2 \dots z_n$, requires no changes in any of the values of the z_i , $i = 1, \dots, n$. Thus, since there are no changes in any of the internal memory variables at a stable combination, a critical race can not occur at a stable combination and can not affect any of the stable combinations of a machine.

♦

We will use the result of Proposition (2.3.1) to show that a critical race does not alter the number of potentially stable states of a machine. First we make a observation on the state assignment of a state with more than one internal variable coding.

Consider the machine $\Sigma = (A, Y, X, f, h)$ of Proposition (2.3.1) and let $x \in X$ be a state of Σ . The state x may have more than one internal coding, i.e. the internal variable representation $z_1 z_2 \dots z_n$ of x may have a set of values all corresponding to the state x in Σ . Suppose that two of these codings differ by more than one internal variable, for instance let $n = 2$ and say x is coded internally by 00 and 11. We could then have a critical race occur in a transition from a state to itself, i.e. $f(00, u) = 11$ and $f(11, u) = 11$. We can ignore the possibility of something like $f(00, u) = 11$ and $f(11, u) = 00$, because this is an unstable cycle which we have excluded from this

discussion. To avoid having to refer to the internal codings, $z_1 z_2 \dots z_n$, instead of the representation of the state, x , in the rest of this work it is assumed that when there are multiple codings of the same state, that any transitions from the state to itself do not involve the change of more than one internal variable. Thus in this discussion a critical race will not appear in a transition from a state to itself. This assumption will not limit any of our results. All the results in this work are valid for a critical race in a transition between a state and itself, but notationally it is much simpler to exclude this case.

(2.3.2) PROPOSITION. Let $\Sigma = (A, Y, X, f, h)$ be a machine and let the pair $(x, u) \in X \times A$ be a critical race in Σ . The critical race (x, u) does not change the number of potentially stable states of Σ .

PROOF. Proposition (2.3.2) is a direct result of Proposition (2.3.1). Recall that for a potentially stable state $x' \in X$, there is an input $u' \in A$ such that $f(x', u') = x'$, qualitatively a potentially stable state x' has at least one stable combination. Since a critical race can not affect a stable combination, the potentially stable states are unaltered. ♦

Proposition (2.3.2) implies that all members of any critical race family of machines have the same number of potentially stable states. This result is necessary for the method of comparison we use between machines.

Next we look at how the stable transition function is affected by a critical race. Let $\Sigma = (A, Y, X, f, h)$ be a machine and let the pair $(x, u) \in X \times A$ be a critical race in Σ . Let $\Sigma|_s = (A, Y, X, s, h)$ be the stable state machine induced by Σ . Pick a potentially stable state $x' \in X$ and an input value $u' \in A$ for which (x', u') is a valid pair. If the next stable state reached by Σ

from (x', u') encounters the critical race (x, u) at some intermediate point, then the point (x', u') is a critical race of the stable state machine $\Sigma_{|s}$. All stable transitions which encounter the critical race at an intermediate point are affected and become critical races in the stable state machine. When Σ is operating in fundamental mode, a critical race (x, u) in Σ may induce up to $\#X$ critical races in $\Sigma_{|s}$, at various points all with the same input value u . Whenever Σ is a direct transition machine, i.e. the stable transition function and the transition function are identical, each critical race in Σ induces one critical race in $\Sigma_{|s}$ at the same point.

CHAPTER 3 CONTROLLERS

3.1 Controllers

As discussed earlier, our objective is to build a controller that will eliminate the effects of a critical race, and we turn now to a somewhat more detailed discussion of controllers. Consider again the sequential machine Σ described by (2.2.1). As can be seen from the figure (1.1), a controller C for Σ has two input variables, the first one for the external reference input v , and the second one for the feedback variable y . To simplify notation, we shall assume that value set of the external reference variable v is identical to the input set A of Σ . Then, the input set of C is the cross product set $A \times Y$, while the output value set of C must be A , since the output of C serves as the input of Σ in the configuration (1.1). We shall denote by Ξ the state set of the controller, and use the letters ϕ and η to denote its transition and output functions, respectively. In other words, the controller is a sequential machine $C = (A \times Y, A, \Xi, \phi, \eta)$.

The machine represented by the closed loop system (1.1) is denoted by Σ_c . In view of our conventions, the input set of Σ_c is A , and its output set is Y , equal to the output set of the original machine Σ . The state set of Σ_c is the cross product set $X \times \Xi$. The transition function of the closed loop system Σ_c is determined by the transition functions of Σ and C , and we shall calculate it later. In view of the fact that the output of the closed loop system

is in fact the output of Σ , the output function of Σ_c is directly determined by the output function h of Σ .

Several issues arise when the control loop is closed around the system Σ in (1.1). The first issue is to guarantee that the closed loop is well posed, i.e., that all signals are uniquely and causally determined by the external reference signal v . A closed loop system, in this case the closed loop system Σ_c , is well posed when either the machine Σ or the controller C is strictly causal and the other is causal. In this work the machine Σ to be controlled is assumed to be strictly causal. Therefore, as long as the controller C is causal, the closed loop system is well posed. This condition is satisfied in our setup, since the machine Σ of (1.1) represents a strictly causal system (see e.g. Hammer [14] for details).

Another important consideration is the requirement that the closure of the control loop around Σ through the controller C not give rise to critical races. To guarantee that no new critical races arise through the closure of the loop, we shall require that the machines Σ and C operate in fundamental mode within the closed loop configuration. To fulfill this requirement, the controller C will be designed to satisfy the following conditions.

- (i) Starting from a stable combination, and in response to a change in the external reference input variable v of C , the output value of the controller does not change until the controller reaches its next stable state.
- (ii) In response to changes in the output y of Σ , the controller C does not commence any state transitions until it identifies that Σ has reached the next stable state.

When these requirements are met, the systems Σ and C will not simultaneously engage in state transitions; furthermore, while one of the

systems Σ or C is engaged in state transitions, its input values will remain fixed, and, for C , only one input variable will change at a time. Of course, the system must be operated so that the rate of changes of the external reference variable v is slow enough, to allow the closed loop system to reach its next stable state before a new input change occurs. This is a standard requirement in the operation of asynchronous sequential machines.

One method of ensuring that the controller output remains fixed during state transitions is to have all controller state transitions be direct transitions. In other words, constructing the controller so that all state transitions go directly to the next stable state without passing through any transitional states.

We are now in a position to formally state the main problem discussed in the present paper. Let $M = \{\Sigma_1, \dots, \Sigma_q\}$ be a critical race family of sequential machines, where each member of M has the input set A , the output set Y , and a state set of cardinality n . For a controller C , denote by Σ_{ic} , the closed loop system with Σ_i inserted for Σ in figure (1.1), and denote by $\Sigma_{ic|s}$ the stable state machine induced by the machine Σ_{ic} . Now, let Σ be a given sequential machine with a stable transition function, having the input set A , the output set Y , and a state set of cardinality n . Then, our main objective is to find a controller C for which the closed loop configuration (1.1) satisfies the condition $\Sigma_{ic|s} = \Sigma$ for all $i = 1, \dots, q$. We shall refer to this problem as the *model matching problem*. A controller C that solves the model matching problem eliminates the effects of the critical race, since the input/output behavior of the closed loop system is always given by the desired model Σ , and does not depend on the outcome of the race.

In the present paper, we consider the model matching problem only for the case where the family M consists of input/state machines, i.e., for the case where the output of each machine Σ_i , $i = 1, \dots, q$, is isomorphic to its state.

Note that in our statement of the model matching problem, the desired model Σ has a state set of cardinality equal to that of the state set of the family M . This requirement reflects our objective of designing a controller that restores a faulty asynchronous machine to an originally intended model. The faulty machine has the same number of states as the intended model.

CHAPTER 4 REACHABILITY

4.1 Reachable and Stably Reachable States

Reachability deals with the characterization of the set of all states that can be reached from a given state in a sequential machine, by applying a list of input characters. In the present section we develop some characterizations of reachability that will play a critical role in our derivation of necessary and sufficient conditions for the existence of controllers that negate the effects of critical races. We start with some basic facts.

Consider a sequential machine Σ described by (2.2.1), and let $x, x' \in X$ be two states of the machine. We say that the state x' is *reachable* from the state x if there is an input list $u = u_0u_1\dots u_k \in A^+$ for which $f(x,u) = x'$, i.e., if the machine can be driven from x to x' . Note that for this definition the list u is required to be non empty. The state x is *controllable* to the state x' whenever x' is reachable from x .

A similar notion can also be defined for the stable state machine $\Sigma|_s$ induced by the machine Σ . Let s be the stable transition function of Σ , and let x and x' be two potentially stable states of Σ . We say that the state x' is *stably reachable* from the state x if there is an input list $u = u_0u_1\dots u_k \in A^+$ for which $x' = s(x,u)$. In view of the definition of the stable transition function s of Σ , the set of stably reachable states is exactly the set of all states that can be reached from x while the machine Σ is operated in

fundamental mode. Again, note that for this definition the list u is required to be non empty. The state x is *stably controllable* to the state x' whenever x' is stably reachable from x . This leads to the following definitions.

A machine $\Sigma = (A, Y, X, f, h)$ is *connected* whenever every state $x \in X$ is reachable from the initial state of Σ , the machine $\Sigma = (A, Y, X, f, h)$ is *stable connected* whenever every state $x \in X$ is stably reachable from the initial state of Σ . A machine $\Sigma = (A, Y, X, f, h)$ is *strongly connected* whenever every state $x \in X$ is reachable from every other state of Σ , the machine $\Sigma = (A, Y, X, f, h)$ is *strongly stable connected* whenever every state $x \in X$ is stably reachable from every other state of Σ .

The following statement is a simple consequence of the fact that the state set X of Σ is of finite cardinality (denoted by $\#X$). It shows that, starting at x , one can reach any stably reachable state in at most $(\#X - 1)$ steps.

(4.1.1) LEMMA. Let Σ be an asynchronous sequential machine of the form (2.2.1), having the state set X and the stable transition function s . Let $x, x' \in X$ be states, and assume that x' is stably reachable from x . Then, there is an input list u of length not exceeding $(\#X - 1)$, for which $x' = s(x, u)$.

PROOF. Since x' is stably reachable from x , there is, by definition, an input list $u' = u_0 u_1 \dots u_k$ such that $x' = s(x, u')$. Now, if $|u'| \leq \#X$, simply set $u := u'$. Otherwise, denote $x_{i+1} := s(x_i, u_i)$, where $x_0 := x$ and $x_{k+1} := x'$, and consider the list of states $x_0, x_1, \dots, x_{k+1}$, which contains $k+2$ elements. Since altogether there are only $\#X$ states and $k+2 > \#X$, at least one state must appear more than once in the list. Let then $i < j$ be integers such that $x_j = x_i$, and consider the input list $w := u_0 u_1 \dots u_{i-1} u_j \dots u_k$ (or

simply $w := u_j \dots u_k$ when $i = 0$). A slight reflection then shows that we have eliminated a cycle, and we still have $x' = s(x, w)$. Clearly, $|w| < |u'|$, and this process can be repeated until an input sequence u satisfying $x' = s(x, u)$ and $|u| \leq \#X - 1$ is obtained. ♦

Consider for a moment an input list $u = u_0 u_1 \dots u_k \in A^+$ for which $x' = s(x, u)$, where s is the stable transition function of the machine Σ . The definition of s implies then that the pair (x', u_k) is a stable combination. Now, let $w \in A$ be any input character for which (x', w) is a stable combination. We can concatenate the input list u with w to obtain the input list $u' := uw$, which still has the property that $x' = s(x, u')$. Thus, the last character of the input list that takes the machine from x to x' can be selected to be any input character that forms a stable combination with the stably reached state x' . This will, however, increase the length of the input list by one step.

In the rest of this work, results are generated using the notion of stably reachable and the stable transition function. In practice, due to timing problems, inputs can only be changed when a machine is in a stable combination, i.e. fundamental mode. The controller, then, can only change inputs to the machine it is controlling when the machine is in a stable combination, and thus the controller can only affect the stable transitions of the machine it is controlling.

4.2 Reachability Sets

Using the notions of reachability and stable reachability, we introduce the notion of a reachability set. Briefly stated, the reachability set of a state x contains all the states reachable from x .

Let $\Sigma = (A, Y, X, f, h)$ be a machine, and let $\#X = n$. Let x be a state of Σ . The *one-step reachability set* $r(x)$ of the state x is the set of all states of Σ reachable from the state x in one step.

$$(4.2.1) \quad r(x) = \{x' \mid x' \in X \text{ and } \exists u \in A \text{ such that } f(x, u) = x'\}$$

We define the *m-step reachability set* $r^m(x)$ as the set of all states of Σ reachable from x in exactly m steps.

$$(4.2.2) \quad r^m(x) = \{x' \mid x' \in X \text{ and } \exists w \in A^+, |w| = m, \text{ such that } f(x, w) = x'\}$$

Now, let $m > 1$ be an integer. In order to determine the states that can be reached in m or less transition steps for the machine Σ , we construct the set

$$(4.2.3) \quad r^{(m)}(x) := \bigcup_{i=1, \dots, m} r^i(x)$$

The set $r^{(m)}(x)$ contains all states reachable from x in m or less steps.

Similarly to Lemma (4.1.1), if there is a state reachable from x in the machine Σ then it is reachable in $(n-1)$ or less steps. This directly implies the following fact.

(4.2.4) PROPOSITION. Let Σ be a sequential machine with a state set consisting of n states. The set of all states reachable from the state x for the machine Σ is the set $r^{(n-1)}(x)$.

We will denote the set of all states reachable from a state x in Σ by the set $r_{\Sigma}(x)$. We will refer to this set as the *reachability set* of the state x .

$$(4.2.5) \quad r_{\Sigma}(x) = \{x' \mid x' \in X \text{ and } \exists w \in A^+ \text{ such that } f(x,w) = x'\}$$

We can now restate our definition of a connected and a strongly connected machine in terms of the reachability set. Consider a machine $\Sigma = (A, Y, X, f, h)$, and let x_0 be the initial state of Σ . The machine Σ is connected whenever $r_{\Sigma}(x_0) = X$. The machine Σ is strongly connected whenever $r_{\Sigma}(x) = X$ for all x in Σ .

A similar notion to the reachability set can be described for the stable state machine $\Sigma|_s$ induced by the machine Σ . Let s be the stable transition function of Σ .

The *one-step stable reachability set* $r_{|s}(x)$ of a potentially stable state x is the set of all potentially stable states of Σ stably reachable from the state x in one step.

$$(4.2.6) \quad r_{|s}(x) = \{x' \mid x' \in X \text{ and } \exists u \in A \text{ such that } s(x,u) = x'\}$$

We define the *m-step stable reachability set* $r_{|s}^m(x)$ as the set of all potentially stable states of Σ stably reachable from a potentially stable state x in exactly m steps.

$$(4.2.7) \quad r_{|s}^m(x) = \{x' \mid x' \in X \text{ and } \exists w \in A^+, |w| = m, \text{ such that } s(x,w) = x'\}$$

Now, let $m > 1$ be an integer. In order to determine the potentially stable states that can be stably reached in m or less stable transition steps for the machine Σ , we construct the set

$$(4.2.8) \quad r_{|s}^{(m)}(x) := \bigcup_{i=1, \dots, m} r_{|s}^i(x)$$

The set $r_{|s}^{(m)}(x)$ contains all potentially stable states which are stably reachable from a potentially stable state x in m or less steps.

Again, in view of Lemma (4.1.1), if there is a state stably reachable from a potentially stable state x in the machine Σ then it is stably reachable in $(n-1)$ or less steps. This leads to the following fact.

(4.2.9) PROPOSITION. Let Σ be a sequential machine with a state set consisting of n states. The set of all states stably reachable from the potentially stable state x for the machine Σ is the set $r_{|s}^{(n-1)}(x)$.

We will denote the set of all potentially stable states stably reachable from a potentially stable state x in Σ by the set $r_{\Sigma|s}(x)$. We will refer to this set as the *stable reachability set* of the potentially stable state x .

$$(4.2.10) \quad r_{\Sigma|s}(x) = \{x' \mid x' \in X \text{ and } \exists w \in A^+ \text{ such that } s(x,w) = x'\}$$

4.3 The Matrix of Stable Transitions

Another useful way of looking at the stable reachability of a machine is through a matrix of inputs for the stable transitions of the machine. We create now a matrix that represents the set of all possible stable transitions for an asynchronous sequential machine Σ . Let $X = \{x_1, \dots, x_n\}$ be the state set of Σ , so that the cardinality $\#X = n$, and let s be the stable transition function of Σ . For every pair of states $x_i, x_j \in X$, let U_{ij} be the set of all input characters $u \in A$ for which $s(x_i, u) = x_j$; i.e., U_{ij} consists of all input characters u that take Σ from the state x_i to the state x_j in one stable transition (although several unstable transitions may occur along the way). Of course, when there is no character u satisfying $s(x_i, u) = x_j$, one has $U_{ij} = \emptyset$, the empty set. Then, the matrix $R(\Sigma)$ of *one-step stable transitions* is an $n \times n$ matrix having the entries R_{ij} as follows: When $U_{ij} \neq \emptyset$, set R_{ij}

$:= U_{ij}$; If $U_{ij} = \emptyset$, write $R_{ij}(\Sigma) := N$ to indicate that there is no one-step stable transition from the state x_i to the state x_j .

Generally speaking, the matrix $R(\Sigma)$ is a matrix whose entries are either sets of words of the input set A , or the extra character N . In other words, the entries of $R(\Sigma)$ are taken from the set $S(A) \cup N$, where $S(A)$ is the set of sentences over the alphabet A . We introduce now a binary operation on such matrices that resembles matrix multiplication. First, some preliminaries. Let U_1 and U_2 be two elements of $S(A) \cup N$. Then, the *concatenation* $\text{conc}(U_1, U_2)$ of U_1 and U_2 is another element of the same set $S(A) \cup N$, given as follows. If either U_1 or U_2 is the character N , we set $\text{conc}(U_1, U_2) := N$. Otherwise, let $u_{11}, \dots, u_{1p}$ be the words of U_1 , and let $u_{21}, \dots, u_{2r}$ be the words of U_2 ; Then, $\text{conc}(U_1, U_2) := \bigcup_{j=1, \dots, p; m=1, \dots, r} u_{1j}u_{2m}$. As we can see, this concatenation operation is non-commutative, and the character N takes the role of a "zero" for this operation. In line with this observation, given a sentence $a \in S(A)$, we define the union $a \cup N := a$ and $N \cup N := N$, so that the character N is omitted in a union with words.

Now, let V and W be two $n \times n$ matrices over the set $S(A) \cup N$, with entries V_{ij} and W_{ij} , $i, j = 1, \dots, n$, respectively. Then, the product $Z := VW$ of V and W is another $n \times n$ matrix over the set $S(A) \cup N$, whose entries Z_{ij} are given by

$$(4.3.1) \quad Z_{ij} := \bigcup_{k=1}^n \text{conc}(V_{ik} W_{kj}), \quad i, j = 1, \dots, n.$$

We can now define powers of the matrix $R(\Sigma)$ in the usual way, by setting

$$(4.3.2) \quad R^m(\Sigma) := R^{m-1}(\Sigma) R(\Sigma)$$

for all integers $m \geq 2$. Then, a slight reflection shows that the (i,j) entry of $R^m(\Sigma)$ consists of all input lists that take the machine Σ from the state x_i to the state x_j in exactly m stable transitions. Note that the actual number of transitions of Σ may be larger than m , since a transition chain may occur along any stable transition. Also, following our convention, the integer m indicates here the number of distinct input values that need to be applied to the machine Σ in order to drive it from x_i to x_j , $i \neq j$, and it does not correspond to the number of steps of the machine. Each input value must be kept constant until the machine reaches its next stable state, so as to obtain fundamental mode operation. Note however that when $i = j$, no changes in the input value are necessary, since the machine starts from a stable combination of the state x_i , and will dwell at the state until an input value that does not form a stable combination with x_i is applied. We shall refer to $R^m(\Sigma)$ as the matrix of *m-step stable transitions*. Here is a simple example for calculating powers of $R(\Sigma)$.

(4.3.3) EXAMPLE. Let a, b, c be characters in the input alphabet A , and consider the matrix

$$R = \begin{pmatrix} \{a,b\} & \{c\} \\ N & \{b,c\} \end{pmatrix}.$$

Then,

$$R^2(\Sigma) = R(\Sigma) R(\Sigma) = \begin{pmatrix} \{aa,ab,ba,bb\} & \{ac,bc,cb,cc\} \\ N & \{bb,bc,cb,cc\} \end{pmatrix} \blacklozenge$$

Now, let $m > 1$ be an integer. In order to determine whether or not the state x_j can be reached from the state x_i in m or less stable transition steps for the machine Σ , we construct the matrix

$$(4.3.4) \quad R^{(m)}(\Sigma) := \bigcup_{i=1, \dots, m} R^i(\Sigma),$$

whose i, j entry is the union of the i, j entries of the matrices $R(\Sigma), \dots, R^m(\Sigma)$. For each $i, j = 1, \dots, n$, the i, j entry of $R^{(m)}(\Sigma)$ consists of all input lists of length m or less that take the machine from the state x_i to the state x_j . Of course, if there is no sequence of stable transitions from x_i to x_j , then entry i, j of $R^{(m)}(\Sigma)$ will be the character N .

Finally, to characterize all possible sequences of stable transitions for the machine Σ , we define the matrix

$$(4.3.5) \quad R^*(\Sigma) := \bigcup_{i=1, \dots, \infty} R^i(\Sigma).$$

In view of Lemma (4.1.1), if there is a sequence of stable transitions that takes the machine Σ from the state x_i to the state x_j , then there is such a sequence of length not exceeding $(n-1)$. This directly implies the following fact.

(4.3.6) PROPOSITION. Let Σ be a sequential machine with a state set consisting of n states, and let $R^{(m)}(\Sigma)$ and $R^*(\Sigma)$ be as defined in (4.3.4) and (4.3.5), respectively. Then, for every pair of integers $i, j \in \{1, \dots, n\}$, the i, j entry of the matrix $R^*(\Sigma)$ is the character N if and only if the i, j entry of the matrix $R^{(n-1)}(\Sigma)$ is the character N .

Thus, the matrix $R^{(n-1)}(\Sigma)$ completely characterizes all pairs of states of the machine Σ among which there are sequences of stable transitions. For this reason, we shall refer to $R^{(n-1)}(\Sigma)$ as the *matrix of stable transitions* of the machine Σ . The matrix of stable transitions contains information that is critical for the construction of controllers that eliminate the effects of races, as we shall discuss later.

We will use the matrix of stable transitions to obtain the appropriate input lists for the construction of corrective controllers. However, for verifying the existence of such controllers, one can use the simpler matrix that we introduce next.

4.4 The Skeleton Matrix

Let Σ be a sequential machine described by (2.2.1), and let $R(\Sigma)$ be the one-step stable transitions matrix of Σ . The *one-step skeleton matrix* $S(\Sigma)$ of Σ is then an $n \times n$ matrix of numbers, whose entries $S_{ij}(\Sigma)$, $i, j = 1, \dots, n$, are all either zero or 1, according to the following rule: If the i, j entry of $R(\Sigma)$ is N , then $S_{ij}(\Sigma) := 0$; otherwise $S_{ij}(\Sigma) := 1$. Thus, the i, j entry of the one-step skeleton matrix is 1 if and only if there is a one-step stable transition that takes Σ from the state x_i to the state x_j . If x_j cannot be reached from x_i in a one step stable transition, then $S_{ij}(\Sigma) = 0$. In simple terms, $S(\Sigma)$ can be regarded as a table that indicates all possible one step stable transitions for the machine Σ .

We can also define powers of the one-step skeleton matrix using the following recursion. Let $m \geq 2$ be an integer, and assume that the $n \times n$ matrix $S^{m-1}(\Sigma)$ has been constructed. Then, $S^m(\Sigma)$ is again an $n \times n$ matrix with entries that are either zero or one, whose entries are given by

$$(4.4.1) \quad S_{ij}^m(\Sigma) = \max \left\{ S_{ik}^{m-1}(\Sigma) S_{kj}(\Sigma), k = 1, \dots, n \right\}, i, j = 1, \dots, n.$$

As we can see, the operation of (4.4.1) resembles matrix multiplication, but, instead of adding the entry products, we calculate the maximum of the entry products, so that each entry is either zero or one. The matrix $S^m(\Sigma)$ is called *m-step skeleton matrix* of the machine Σ . The m -step skeleton matrix is related to the matrix $R^m(\Sigma)$ constructed in the

previous subsection in the same way that $S(\Sigma)$ is related to $R(\Sigma)$; i.e., an entry i, j of $S^m(\Sigma)$ is non zero if and only if it is possible to reach the state x_j from the state x_i in m stable transitions. This is the subject of the next statement.

(4.4.2) PROPOSITION. Let Σ be a sequential machine with n states, and let $m \geq 1$ be an integer. Let $R^m(\Sigma)$ be the matrix of m -step stable transitions of Σ , and let $S^m(\Sigma)$ be the m -step skeleton matrix of Σ . Then, for all $i, j \in \{1, \dots, n\}$, the entry i, j entry of $S^m(\Sigma)$ is non zero if and only if the i, j entry of $R^m(\Sigma)$ is not N .

PROOF. The statement holds for $m = 1$ by the definition of the one-step skeleton matrix $S(\Sigma)$. In preparation for an induction, assume the Proposition is true for $m = r$, where $r \geq 1$ is an integer. We now show that this implies that the Proposition holds for $m = r+1$. Consider an arbitrary entry i, j of the matrix $R^m(\Sigma)$, where $i, j \in \{1, \dots, n\}$. In view of (4.3.2) and (4.3.1), the i, j entry $R_{ij}^{r+1}(\Sigma)$ of $R^{r+1}(\Sigma)$ is N if and only if the following condition holds: For each integer $k = 1, \dots, n$, at least one of the two entries $R_{ik}^r(\Sigma)$ or $R_{kj}(\Sigma)$ is equal to N . In view of our induction assumption, this implies that the entry $R_{ij}^{r+1}(\Sigma)$ is N if and only if the following condition holds: For each integer $k = 1, \dots, n$, at least one of the two entries $S_{ik}^r(\Sigma)$ or $S_{kj}(\Sigma)$ is equal to zero. But by (4.4.1), this last condition is equivalent to the condition $S_{ij}^{r+1}(\Sigma) = 0$. Summarizing, we have obtained that $S_{ij}^{r+1}(\Sigma) = 0$ if and only if $R_{ij}^{r+1}(\Sigma)$ is N . ♦

We define now the *skeleton matrix* $S^{(n-1)}(\Sigma)$ of the machine Σ as a $n \times n$ matrix of zeros and ones, whose i, j entry $S_{ij}^{(n-1)}(\Sigma)$ is given by

$$(4.4.3) \quad S_{ij}^{(n-1)}(\Sigma) := \max_{m=1, \dots, n-1} S_{ij}^m.$$

Here, n is the number of states of the machine Σ . In view of Proposition (4.4.2), the skeleton matrix $S^{(n-1)}(\Sigma)$ corresponds to the matrix $R^{(n-1)}(\Sigma)$ of stable transitions in the following way. For every pair of integers $i, j \in \{1, \dots, n\}$, the entry $S^{(n-1)}_{ij}(\Sigma)$ is zero if and only if the corresponding entry $R^{(n-1)}_{ij}(\Sigma)$ is N . Recalling the significance of the matrix $R^{(n-1)}(\Sigma)$ of stable transitions, it follows that the skeleton matrix characterizes the pairs of all states of Σ among which one can move via stable transitions, in any number of steps. Explicitly, it is possible to move from a state x_i to a state x_j of the machine Σ via a sequence of stable transitions (of unspecified length) if and only if the i, j entry of the skeleton matrix $S^{(n-1)}(\Sigma)$ is equal to one. This justifies the following statement.

(4.4.4) PROPOSITION. Let Σ be a sequential machine having the state set $X = \{x_1, \dots, x_n\}$ and the skeleton matrix $S^{(n-1)}(\Sigma)$, and let $i, j \in \{1, \dots, n\}$ be two integers. Then, the state x_j is stably reachable from the state x_i if and only if the entry i, j of the skeleton matrix $S^{(n-1)}(\Sigma)$ is equal to one.

4.5 Reachability for Families of Machines

As we have discussed previously, the main subject of the present paper revolves around families of sequential machines, rather than sequential machines as individual entities. The skeleton matrix will help us identify the set of stable transitions that are common to all members of a critical race family of sequential machines. This set is crucial for the determination of whether or not a controller can be found to eliminate the adverse effects of a critical race.

Consider a family $M = \{\Sigma_1, \dots, \Sigma_q\}$ of sequential machines, all of which have the same state space $X = \{x_1, \dots, x_n\}$, and let $i, j \in \{1, \dots, n\}$ be two integers. We say that a state x_j is *stably reachable within the family* M from the state x_i if the state x_j is stably reachable from x_i in each one of the machines $\Sigma_1, \dots, \Sigma_q$. We comment that when x_j is stably reachable from x_i in the family M , the sequence of input characters that take a machine of the family M from x_i to x_j may differ from one member of the family M to another.

We can form the stable reachability set for the family of machines M by

$$(4.5.1) \quad r_{M|s}(x) := \bigcap_{i=1, \dots, q} r_{\Sigma_i|s}(x).$$

The set $r_{M|s}(x)$ contains all states stably reachable from the state x in the family of machines M .

Still referring to the family $M = \{\Sigma_1, \dots, \Sigma_q\}$ of machines, let $S^{(n-1)}(\Sigma_i)$ be the skeleton matrix of the machine Σ_i , $i = 1, \dots, q$. Then, the *skeleton matrix* $\mathcal{S}(M)$ of the family M is defined as an $n \times n$ matrix of zeros and ones, whose entries $\mathcal{S}_{ij}(M)$ are given by

$$(4.5.2) \quad \mathcal{S}_{ij}(M) := \min_{i=1, \dots, q} S^{(n-1)}(\Sigma_i).$$

It follows then directly from Proposition (4.4.4) that the skeleton matrix of the family M characterizes the set of all pairs of states that are stably reachable from each other within the family M , as follows.

(4.5.3) PROPOSITION. Let $M = \{\Sigma_1, \dots, \Sigma_q\}$ be a family of sequential machines, all of which have the state space $X = \{x_1, \dots, x_n\}$, and let $\mathcal{S}(M)$ be the skeleton matrix of the family M . Then, for any pair of integers $i, j \in \{1, \dots, n\}$, the state x_j is stably reachable from the state x_i within the family M if and only if the i, j entry of $\mathcal{S}(M)$ is equal to one.

CHAPTER 5 EXISTENCE OF CORRECTIVE CONTROLLERS

In the present chapter we derive necessary and sufficient conditions for the existence of controllers that eliminate the effects of a critical race, and assign desirable properties to a given sequential machine Σ . Recall that the controller is a sequential machine C that is connected to the machine Σ according to the closed loop configuration (1.1).

5.1 Operation of Corrective Controllers

A controller can ensure all stable transitions of the closed loop system (1.1) are deterministic. Let $\Sigma|_s = (A, Y, X, s, h)$ be the stable state machine induced by a machine $\Sigma = (A, Y, X, f, h)$. There is a controller C such that the closed loop system Σ_c of (1.1) is deterministic. Let Σ be race free, a controller C which passes the reference input, unmodified, to Σ ensures the closed loop system Σ_c remains deterministic. Next let Σ have a critical race which induces k critical races in the stable state machine $\Sigma|_s$. Denote the critical race pairs in $\Sigma|_s$ by (x_i, u_i) , $i = 1, \dots, k$. A controller C can be constructed to pass any input except u_i whenever $\Sigma|_s$ is in a stable combination of the state x_i , $i = 1, \dots, k$. The controller passes the reference input, unmodified, when $\Sigma|_s$ is in any other state.

Consider two machines $\Sigma = (A, Y, X, f, h)$ and $\Sigma' = (A, Y, X, f, h)$ that have the same input sets, the same output sets, the same state set, and the

same output function. Recall Σ is equivalent to Σ' whenever for all $x \in X$ and all permissible input lists $a \in A^+$, the same output sequence is generated by Σ and Σ' whenever the list a is applied starting in the state x . Here, Σ and Σ' have the same input and output sets, as well as the same output function h , thus Σ is equivalent to Σ' whenever the transition functions f and f' are identical, i.e. the sequences of states Σ and Σ' pass through when driven by the input list a are identical. Similarly the stable state machine $\Sigma|_s$ induced by Σ is equivalent to the stable state machine $\Sigma|_s'$ induced by Σ' whenever the stable transition function s of $\Sigma|_s$ is identical to the stable transition function s' of $\Sigma|_s'$.

Now consider the closed loop system $\Sigma_c = (A, Y, X \times \Xi, g, h)$ consisting of the machine $\Sigma = (A, Y, X, f, h)$ with a controller $C = (A \times Y, Y, \Xi, \phi, \eta)$ connected as figure (1.1). The transition function g of Σ_c is defined by $g((x, \alpha), v) = (f(x, \eta(\alpha, (x, v))), \phi(\alpha, (x, v)))$ for all valid $(x, \alpha) \in X \times \Xi$ and $v \in A$. For the model matching problem, we must find a controller C such that $\Sigma_{c|s}$ is equivalent to $\Sigma|_s'$. The state set of Σ_c is not the same as the state set of Σ' , however, the output functions of Σ_c and Σ' are both h , and h depends only on the states from X not $X \times \Xi$. The closed loop system $\Sigma_{c|s}$ is equivalent to $\Sigma|_s'$ whenever the stable transition function γ of $\Sigma_{c|s}$ satisfies $\gamma((x, \alpha), v) = (s'(x, v), \alpha')$ for all valid $(x, \alpha) \in X \times \Xi$ and $v \in A$, and some $\alpha' \in \Xi$. In other words, the next stable state of $\Sigma_{c|s}$ initially in (x, α) receiving the input value v has its element from X equal to the next stable state of $\Sigma|_s'$ initially in x receiving the input value v .

Before discussing the necessary and sufficient conditions for the existence of a corrective controller, we discuss the changes a controller can make to a machine. We begin by discussing the changes that can be made to a single stable transition of a machine. This leads to the following statement.

(5.1.1) LEMMA. Let $\Sigma_{|s} = (A, Y, X, s, h)$ be the stable state machine induced by a machine $\Sigma = (A, Y, X, f, h)$. Choose a state $x_1 \in X$ and a set of input values $A_1 \subset A$ which form valid pairs with the state x_1 . Let $x_1' \in X$ be any state stably reachable from x_1 . Then, there is a controller C such that the stable state closed loop system $\Sigma_{c|s}$ is equivalent to a machine Σ' that has the following properties:

- (i) $\Sigma' = (A, Y, X, s', h)$ has the same input, output, and state sets, as well as the same output function, as the machine $\Sigma_{|s}$.
- (ii) The transition function s' of Σ' is equal to the transition function s of $\Sigma_{|s}$ at all but the pairs (x_1, A_1) , at which it takes the value x_1' , i.e. $s'(z, u) = s(z, u)$ for all $(z, u) \in X \times A \setminus (x_1, A_1)$, and $s'(x_1, A_1) = x_1'$.

PROOF. By the choice of x_1' there is an input list $w_1 \in A^+$, $w_1 = v_0 v_1 \dots v_{m-1}$, such that $s(x_1, w_1) = x_1'$. Denote $x_{11} := s(x_1, v_0)$, $x_{12} := s(x_{11}, v_1), \dots, x_{1(m-1)} := s(x_{1(m-2)}, v_{m-2})$, $x_1' := s(x_{1(m-1)}, v_{m-1})$, $x_{1i} \in X$ for $i = 1, \dots, m-1$.

In intuitive terms the controller we build functions as follows. The controller stays in its initial state α_0 until a stable combination of the state x_1 occurs. When a stable combination of x_1 occurs, the controller enters the state α_1 . The controller returns to its initial state from the state α_1 when its input is any pair except stable combinations of x_1 or one of the pairs (x_1, A_1) . The controller stays in its state α_1 whenever the input to the controller is a stable combination of x_1 . The controller enters its state α_2 from the state α_1 whenever the input to the controller is one of the pairs (x_1, A_1) . At the state α_2 the controller produces the output v_0 . It remains in that state until the machine reaches the pair (x_{11}, v_0) ; i.e. until the machine reaches its next stable combination. When (x_{11}, v_0) occurs, the controller goes to the state α_3 , producing the output value v_1 . This

continues for $2 \leq i \leq m$ with at each state α_i the controller produces the output value v_{i-2} until the state $x_{1(i-1)}$ is reached. Then the controller goes to the next state α_{i+1} . When the state $x_{1(m-1)}$ is reached, the controller will enter the state α_{m+1} . The controller produces the output value v_{m-1} until the state x_1' is reached by the machine. The controller returns to the state α_0 once the reference input is changed.

The stable transition function γ of the closed loop system Σ_c satisfies $\gamma(x_1, \alpha_0, A_1) = (x_1', \alpha_{m+1})$ and $\gamma(z, \alpha_0, u) = (s(z, u), \alpha_0)$ for all $(z, u) \in X \times A \setminus (x_1, A_1)$.

We now construct a controller $C = (A \times Y, A, \Xi, \phi, \eta)$ as follows. The state set Ξ of the controller consists of $m+2$ states, which we denote by the symbols $\alpha_0, \alpha_1, \dots, \alpha_{m+1}$. We define the transition function ϕ from the state α_0 by setting $\phi(\alpha_0, (z, u)) = \alpha_0$ for all $(z, u) \in X \times A \setminus (x_1, U_1)$ where $U_1 \subset A$ is the set of input characters which form stable combinations with the state x_1 , and $\phi(\alpha_0, (x_1, U_1)) = \alpha_1$. The value of the output function η for the same state is $\eta(\alpha_0, (z, u)) = u$ for all $(z, u) \in X \times A$.

The transition function from the state α_1 is defined by setting $\phi(\alpha_1, (x_1, A_1)) = \alpha_2$, $\phi(\alpha_1, (x_1, U_1)) = \alpha_1$ where $U_1 \subset A$ is the set of input characters which form stable combinations with the state x_1 , and $\phi(\alpha_1, (z, u)) = \alpha_0$ for all other $(z, u) \in X \times A$. The value of the output function at the same state is $\eta(\alpha_1, (z, u)) = u_1$ where $u_1 \in U_1$.

The rest of the transition function and output function are given by:
 $\phi(\alpha_2, (z, u)) = \alpha_2$ for all $(z, u) \in X \times A \setminus (x_{11}, A_1)$ and $\phi(\alpha_2, (x_{11}, A_1)) = \alpha_3$
 $\eta(\alpha_2, (z, u)) = v_0$ for all $(z, u) \in X \times A$
 $\phi(\alpha_3, (z, u)) = \alpha_3$ for all $(z, u) \in X \times A \setminus (x_{12}, A_1)$ and $\phi(\alpha_3, (x_{12}, A_1)) = \alpha_4$
 $\eta(\alpha_3, (z, u)) = v_1$ for all $(z, u) \in X \times A$
 for $4 < i < m$

$\phi(\alpha_i, (z, u)) = \alpha_i$ for all $(z, u) \in X \times A \setminus (x_{1(i-1)}, A_1)$ and $\phi(\alpha_i, (x_{1(i-1)}, A_1)) = \alpha_{i+1}$

$\eta(\alpha_i, (z, u)) = v_{i-2}$ for all $(z, u) \in X \times A$

and for $i = m+1$

$\phi(\alpha_{m+1}, (z, u)) = \alpha_{m+1}$ for all $(z, u) \in X \times A \setminus (x_1', A - A_1)$ and $\phi(\alpha_{m+1}, (x_1', A - A_1)) = \alpha_0$.

$\eta(\alpha_{m+1}, (z, u)) = v_{m-1}$ for all $(z, u) \in X \times A$.

We can see that the output of the controller C to the machine Σ is the input to C whenever any pair except one of (x_1, A_1) occurs. The controller stays in its initial state α_0 and outputs the current input to the machine Σ until a stable combination of the state x_1 occurs. On detecting a stable combination of x_1 the controller goes to the state α_1 . In the state α_1 the controller looks for one of the pairs (x_1, A_1) . If one of the pairs (x_1, A_1) is detected after a stable combination of the state x_1 , the controller outputs the list w_1 to the machine Σ which brings Σ to the desired state x_1' . Once Σ has reached the state x_1' the controller waits until the reference input is changed and then reenters its initial state α_0 .

Now assume there is a controller C such that the closed loop system Σ_{cls} is equivalent to Σ' and that x_1' is not stably reachable from x_1 . Recall that Σ_{cls} is equivalent to Σ' whenever $\gamma((x, \alpha), u) = (s'(x, u), \alpha')$ for all valid pairs $(x, u) \in X \times A$ and controller states $\alpha, \alpha' \in \Xi$, and that $s'(x, u) = s(x, u)$ for all valid pairs $(x, u) \in X \times A$ except (x_1, A_1) for which $s'(x_1, A_1) = x_1'$. A controller C , therefore, must ensure that Σ_{cls} initially in the state x_1 receiving one of the input values A_1 has x_1' as its next stable state.

Also recall that the only access a controller C has to a machine Σ is through the input to Σ . A controller C can only cause a stable transition from x_1 to x_1' in Σ by providing an appropriate input list $w \in A^+$ to Σ which drives Σ from x_1 to x_1' . This implies that there is a $w \in A^+$ such

that $s(x_1, w) = x_1'$, and that C provides the list w to Σ when one of the points (x_1, A_1) occurs. However, the state x_1' is not stably reachable from x_1 , so there is no $w \in A^+$ such that $s(x_1, w) = x_1'$. In other words, a stable transition from x_1 to x_1' does not exist in Σ . There is no possible way for the machine Σ to enter the state x_1' starting from the state x_1 operating in fundamental mode. This is a contradiction, and x_1' must be stably reachable from x_1 whenever a controller C exists so the closed loop system $\Sigma_{c|s}$ is equivalent to Σ' . ♦

The controller provides an input list which drives the machine along a path ending at the desired combination. The input list is provided one character at a time, and each new character is given when the machine arrives in its next stable combination from the previous input character. This continues until the entire input list is used and the machine reaches the desired stable combination. The controller, in effect, substitutes a different desired stable transition for the original stable transition.

The controller C from Lemma (5.1.1) has $m+2$ states, where m is the length of the path from the initial pair to the desired end pair. There are an infinite number of controllers C' with more states than C that effect the same changes on Σ as the controller C . There may also be controllers C'' with less states than C that effect the same changes in Σ as the controller C . Minimization procedures exist which can provide a corrective controller C with the smallest number of states (see Kohavi [24] for details).

We denote the controller of Lemma (5.1.1) by $C((x_1, A_1), w_1, x_1')$. The controller $C((x_1, A_1), w_1, x_1')$ provides the input list w_1 to the machine it is controlling whenever the controller recognizes one of the pairs (x_1, A_1)

after a stable combination of the state x_1 . This will take the machine being controlled to the state x_1' .

Recall that Σ is an input/state machine, the output of Σ is isomorphic to the current state of Σ , $h(x,u) = \Psi_h(x)$ for all valid $(x,u) \in X \times A$. If the output isomorphism Ψ_h is not the identity map, the input to the controller needs to be precompensated. The controller will need to receive $\Psi_h^{-1}(h(x,u)) = x$. Including the precompensator, we can write the controller in the form $C = (A \times \Psi_h^{-1}Y, A, \Xi, \phi, \eta)$. In the present work, we will consider machines whose output is the current state, i.e. Ψ_h is the identity map.

We stated earlier that a corrective controller C connected as in figure (1.1) with a machine Σ ensures that the stable state closed loop system $\Sigma_c|_S$ operates in fundamental mode, i.e. the addition of a controller C to a machine Σ does not create any additional critical races. Referring to the controller of Lemma (5.1.1), we show this result now.

(5.1.2) PROPOSITION. The closed loop system Σ_c of Lemma (5.1.1) operates in fundamental mode.

PROOF. Let Σ_c start in one of its stable combinations, $((x, \alpha_0), v)$.

Responding to a change in the reference signal, the controller receives a pair, say (x, u) . From its state α_0 , the controller remains in α_0 whenever the input pair (x, u) is not one of the stable combinations of the state x_1 . There is no state change in C , so the output of C may be changed in α_0 . In α_0 , the reference input value is the output to the machine Σ , and Σ_c operates in fundamental mode (C makes no changes in state, so Σ is allowed to be changing).

Also from its state α_0 , whenever the input pair is a stable combination of the state x_1 , the controller C enters its next stable state α_1 .

The output of C remains constant (at the previous reference input value) until the state α_1 is reached. Once α_1 is reached, the controller may change its output value to one of the values that form a stable combination with the state x_1 . We can see that while C changes state, the output value of C remains constant. Thus Σ will not start any changes until C has reached its next stable state α_1 , and changes its output value. Thus Σ_c operates in fundamental mode whenever C is in its state α_0 .

Now suppose the controller C is in its state α_1 . For this to occur, Σ must be in a stable combination of the state x_1 . When in the state α_1 , the controller outputs a value which forms a stable combination with x_1 . The controller remains in the state α_1 and its output value remains the same unless one of the pairs (x_1, A_1) occurs or the reference input is changed to a value in $A - (A_1 \cup U_1)$. When one of the pairs (x_1, A_1) occurs, the controller enters its next stable state α_2 . The output remains fixed at the previous value until α_2 is reached by C . When a reference input in $A - (A_1 \cup U_1)$ occurs the controller enters its initial state α_0 . Again the controller output remains fixed at the previous value until α_0 is reached. Thus the controller output does not change until any state transitions of the controller C are finished, and Σ_c operates in fundamental mode whenever C is in its state α_1 .

Now for the rest of the states of the controller $\alpha_i, i = 2, \dots, m$, the closed loop systems operates as follows. In the state α_i , the controller C outputs the value v_{i-2} . The controller remains in the state α_i until the machine Σ reaches its next stable state $x_{1(i-1)}$. When Σ reaches $x_{1(i-1)}$ the controller enters the state α_{i+1} . The output of the controller remains v_{i-2} until C has reached α_{i+1} . Thus at each state α_i , the controller output remains constant while Σ is changing states, and Σ is in a stable

combination whenever C is changing states. Thus Σ_c operates in fundamental mode when the controller is in the states α_i , $i = 2, \dots, m$.

Finally, for the controller state α_{m+1} , the controller output remains constant, v_{m-1} , until Σ reaches x_1' . Once Σ has reached Σ' , the reference input is allowed to change, and the controller reenters its initial state α_0 . The controller output remains v_{m-1} until α_0 is reached. Thus the closed loop system Σ_c operates in fundamental mode for all states of the controller, and as long as the reference inputs do not violate fundamental mode, neither will the system Σ_c .

In more intuitive terms, in response to a change in the output of C , the machine Σ may commence a series of transitions. We can see from the controller of Lemma (5.1.1) that the controller C does not commence any state transitions until it identifies that Σ has reached its next stable state in response to the change in input. Thus the output and state of C are kept constant while Σ is changing states. The output of C is also kept constant while C is changing states ensuring that Σ remains constant during any controller state transitions. In this way Σ and C will not simultaneously engage in state transitions; furthermore, while one of the systems Σ or C is engaged in state transitions, its input values will remain fixed. Of course, the system must be operated so that the rate of changes of the external reference variable v is slow enough, to allow the closed loop system to reach its next stable state before a new input change occurs. This is a standard requirement in the operation of asynchronous sequential machines. ♦

Next we extend the result of Lemma (5.1.1) to any number of stable transitions of a machine.

(5.1.3) THEOREM. Let $\Sigma|_S = (A, Y, X, s, h)$ be the stable state machine induced by a machine $\Sigma = (A, Y, X, f, h)$. Choose k states $x_i \in X, i = 1, \dots, k$, and k sets of input values $A_i \subset A$, where A_i is a set of inputs which form valid pairs with the state $x_i, i = 1, \dots, k$. Let $x_i' \in X$ be any state stably reachable from $x_i, i = 1, \dots, k$. Then, there is a controller C such that the stable state closed loop system $\Sigma_{c|S}$ is equivalent to a machine Σ' that has the following properties:

- (i) $\Sigma' = (A, Y, X, s', h)$ has the same input, output, and state sets, as well as the same output function, as the machine $\Sigma|_S$.
- (ii) The transition function s' of Σ' is equal to the transition function s of $\Sigma|_S$ at all but the pairs (x_i, A_i) at which it takes the value $x_i', i = 1, \dots, k$. i.e. $s'(z, u) = s(z, u)$ for all $(z, u) \in X \times A \setminus (x_i, A_i)$ and $s'(x_i, A_i) = x_i', i = 1, \dots, k$.

PROOF. First all items indexed the same correspond to each other, i.e. $(x_i, A_i), w_i, x_i'$ all correspond to each other.

By the choice of x_i' there is a $w_i \in A^+, w_i = v_{i0}v_{i1}\dots v_{i(m(i)-1)}$ such that $s(x_i, w_i) = x_i'$. Denote $x_{i1} := s(x_i, v_{i0}), x_{i2} := s(x_{i1}, v_{i1}), x_{i3} := s(x_{i2}, v_{i2}), \dots, x_{i(m(i)-1)} := s(x_{i(m(i)-2)}, v_{i(m(i)-2)}), x_i' := s(x_{i(m(i)-1)}, v_{i(m(i)-1)}), i = 1, \dots, k$.

We construct a controller $C = (A \times Y, A, \Xi, \phi, \eta)$ as follows. The state set of the controller Ξ consists of $2 + \sum_{i=1}^k |w_i| = 2 + \sum_{i=1}^k m(i)$ states denoted by Ξ

=

$\{\alpha_0, \alpha_1, \alpha_{11}, \alpha_{12}, \dots, \alpha_{1m(1)}, \alpha_{21}, \dots, \alpha_{2m(2)}, \dots, \alpha_{i1}, \dots, \alpha_{im(i)}, \dots, \alpha_{k1}, \dots, \alpha_{km(k)}\}$. We define the transition function ϕ from the state α_0 by setting $\phi(\alpha_0, (z, u)) = \alpha_0$ for all $(z, u) \in X \times A \setminus (x_i, U_i)$ where $U_i \subset A$ is the set of input values which form stable combinations with the state x_i , and $\phi(\alpha_0, (x_i, U_i)) = \alpha_1, i = 1, \dots, k$.

The value of the output function η for the same state is $\eta(\alpha_0, (z, u)) = u$ for all $(z, u) \in X \times A$.

The transition function from the state α_1 is defined by setting $\phi(\alpha_1, (x_i, A_i)) = \alpha_{i1}$, $\phi(\alpha_1, (x_i, U_i)) = \alpha_1$, $i = 1, \dots, k$ and $\phi(\alpha_1, (z, u)) = \alpha_0$ for all other $(z, u) \in X \times A$. The value of the output function at the same state is $\eta(\alpha_1, (z, u)) = u_i$ where $u_i \in U_i$. The rest of the transition function ϕ and output function η are given by:

for $i = 1, \dots, k$

$\phi(\alpha_{i1}, (z, u)) = \alpha_{i1}$ for all $(z, u) \in X \times A \setminus (x_{i1}, A_i)$ and $\phi(\alpha_{i1}, (x_{i1}, A_i)) = \alpha_{i2}$

$\eta(\alpha_{i1}, (z, u)) = v_{i0}$ for all $(z, u) \in X \times A$

$\phi(\alpha_{i2}, (z, u)) = \alpha_{i2}$ for all $(z, u) \in X \times A \setminus (x_{i2}, A_i)$ and $\phi(\alpha_{i2}, (x_{i2}, A_i)) = \alpha_{i3}$

$\eta(\alpha_{i2}, (z, u)) = v_{i1}$ for all $(z, u) \in X \times A$

for $3 \leq j < m(i)-1$

$\phi(\alpha_{ij}, (z, u)) = \alpha_{ij}$ for all $(z, u) \in X \times A \setminus (x_{ij}, A_i)$ and $\phi(\alpha_{ij}, (x_{ij}, A_i)) = \alpha_{ij+1}$

$\eta(\alpha_{ij}, (z, u)) = v_{ij-1}$ for all $(z, u) \in X \times A$

and for $j = m(i)$

$\phi(\alpha_{im(i)}, (z, u)) = \alpha_{im(i)}$ for all $(z, u) \in X \times A \setminus (x_i', A - A_i)$ and $\phi(\alpha_{im(i)}, (x_i', A - A_i)) = \alpha_0$

$\eta(\alpha_{im(i)}, (z, u)) = v_{i(m(i)-1)}$ for all $(z, u) \in X \times A$

The output of the controller C to the machine Σ is the reference input to C whenever any combination except one of the pairs (x_i, A_i) , $i = 1, \dots, k$ occur. The controller stays in its initial state α_0 and outputs the current input to the machine Σ until one of the stable combinations of the states x_i , $i = 1, \dots, k$ occur. On detecting one of the stable combinations of one of the states x_i , $i = 1, \dots, k$ the controller goes to the state α_1 . In the state α_1 the controller looks for one of the pairs (x_i, A_i) , $i = 1, \dots, k$. If one of the pairs (x_i, A_i) is detected after a stable combination of x_i , the controller

outputs the appropriate list w_i to the machine Σ which brings Σ to the desired state x_i' . Once Σ has reached the state x_i' , the controller waits until the reference input is changed and then reenters its initial state α_0 .

Now assume there is a controller C such that the closed loop system $\Sigma_{c|s}$ is equivalent to Σ' and that x_i' is not stably reachable from x_i for some $i, i = 1, \dots, k$. Recall that $\Sigma_{c|s}$ is equivalent to Σ' whenever $\gamma((x, \alpha), u) = (s'(x, u), \alpha')$ for all valid pairs $(x, u) \in X \times A$ and controller states $\alpha, \alpha' \in \Xi$, and that $s'(x, u) = s(x, u)$ for all valid pairs $(x, u) \in X \times A$ except (x_i, A_i) for which $s'(x_i, A_i) = x_i', i = 1, \dots, k$. A controller C must ensure that $\Sigma_{c|s}$ initially in one of the states x_i receiving one of the input values A_i has x_i' as its next stable state, $i = 1, \dots, k$. Also recall that the only access a controller C has to a machine Σ is through the input to Σ . The corrective controller C can cause a transition from x_i to x_i' in Σ by providing an appropriate input list $w_i \in A^+$ to Σ which drives Σ from x_i to $x_i', i = 1, \dots, k$. This implies that for every $i, i = 1, \dots, k$, there is a $w_i \in A^+$ such that $s(x_i, w_i) = x_i'$, and that C provides the list w_i to Σ when one of the points (x_i, A_i) occurs.

However, if any x_i' is not stably reachable from $x_i, i = 1, \dots, k$, then there is no $w_i \in A^+$ such that $s(x_i, w_i) = x_i'$. So there is at least one set of pairs (x_i, A_i) for which a controller cannot provide an appropriate input list for $\gamma((x_i, \alpha), A_i) = (x_i', \alpha')$. This is a contradiction, and x_i' must be stably reachable from x_i for all $i = 1, \dots, k$ whenever a controller C exists such that $\Sigma_{c|s}$ is equivalent to Σ' . ♦

The controller C from Theorem (5.1.3) has $2 + \sum_{i=1}^k |m(i)|$ states,

where $m(i)$ is the length of the path from the i th starting pair to the

desired end pair. There are an infinite number of controllers C' with more states than C that effect the same changes in Σ as the controller C . There may also be controllers C'' with less states than C that effect the same changes in Σ as the controller C .

The controller designed in Theorem (5.1.3) is similar to combining k parallel controllers like the controller of Lemma (5.1.1). Only one of the parallel components is ever active at any one time, and each component is independent of all the other components. We denote the controller of Theorem (5.1.3) by $C(\{(x_1, A_1), (x_2, A_2), \dots, (x_k, A_k)\}, \{w_1, \dots, w_k\}, \{x_1', \dots, x_k'\})$ which we can shorten to $C((x_i, A_i), w_i, x_i'), i = 1, \dots, k$. The controller $C((x_i, A_i), w_i, x_i'), i = 1, \dots, k$ provides the input list w_i to the machine it is controlling whenever the controller recognizes one of the pairs $(x_i, A_i), i = 1, \dots, k$ after a stable combination of one of the states $x_i, i = 1, \dots, k$. This will take the machine being controlled to the state x_i' .

We can see that the controller constructed in Theorem (5.1.3) connected with a machine Σ as in (1.1) operates in fundamental mode. The argument follows that of Proposition (5.1.2).

Let $\Sigma = (A, Y, X, f, h)$ be a machine, and denote the states of Σ by $X = \{x_1, \dots, x_n\}$. Now recall from the definition of the skeleton matrix that whenever the i, j entry of the skeleton matrix $S^{(n-1)}(\Sigma)$ of the machine Σ is a one then the state x_j of Σ is stably reachable from the state x_i of Σ . Using this result we can restate Lemma (5.1.1) and Theorem (5.1.3).

(5.1.4) PROPOSITION. Let $\Sigma|_s = (A, Y, X, s, h)$ be the stable state machine induced by a machine $\Sigma = (A, Y, X, f, h)$ and let $\#X = n$. Choose a state $x_1 \in X$ and a set of input values $A_1 \subset A$ which form valid pairs with the state x_1 . Let $x_1' \in X$ be any state such that the entry of the skeleton matrix $S^{(n-1)}$

${}^1(\Sigma)$ indexed by the states x_1 and x_1' is a one. Then, there is a controller C such that the stable state closed loop system $\Sigma_{c|s}$ is equivalent to a machine Σ' that has the following properties:

- (i) $\Sigma' = (A, Y, X, s', h)$ has the same input, output, and state sets, as well as the same output function, as the machine $\Sigma_{|s}$.
- (ii) The transition function s' of Σ' is equal to the transition function s of $\Sigma_{|s}$ at all but the points (x_1, A_1) , at which it takes the value x_1' , i.e. $s'(z, u) = s(z, u)$ for all $(z, u) \in X \times A \setminus (x_1, A_1)$, and $s'(x_1, A_1) = x_1'$.

(5.1.5) PROPOSITION. Let $\Sigma_{|s} = (A, Y, X, s, h)$ be the stable state machine induced by a machine $\Sigma = (A, Y, X, f, h)$ and let $\#X = n$. Choose k states $x_i \in X$, $i = 1, \dots, k$, and k sets of input values $A_i \subset A$ where A_i is a set of inputs which form valid pairs with the state x_i . Let $x_i' \in X$ be any state such that the entry of the skeleton matrix $S^{(n-1)}(\Sigma)$ indexed by the states x_i and x_i' is a one, $i = 1, \dots, k$. Then, there is a controller C such that the stable state closed loop system $\Sigma_{c|s}$ is equivalent to a machine Σ' that has the following properties:

- (i) $\Sigma' = (A, Y, X, s', h)$ has the same input, output, and state sets, as well as the same output function, as the machine $\Sigma_{|s}$.
- (ii) The transition function s' of Σ' is equal to the transition function s of $\Sigma_{|s}$ at all but the points (x_i, A_i) , at which it takes the value x_i' , $i = 1, \dots, k$. i.e. $s'(z, u) = s(z, u)$ for all $(z, u) \in X \times A \setminus (x_i, A_i)$ and $s'(x_i, A_i) = x_i'$, $i = 1, \dots, k$.

Whenever a stable state machine $\Sigma_{|s}$ is strongly connected, the desired end state x_i' can be any potentially stable state of $\Sigma_{|s}$.

The necessary input lists for the controller design are found in the appropriate entries of the matrix of stable transitions $R^{(n-1)}(\Sigma)$. The input

lists, however, are not always minimal. There may be stable cycles, which can be removed, and the resulting minimal input list is used to build the corrective controller.

The results of Theorem (5.1.3) and Proposition (5.1.5) explain the changes a controller can effect on a machine. We proceed now to finding when a corrective controller C exists for a machine Σ , such that the stable state closed loop system $\Sigma_{c|s}$ is equivalent to a machine Σ' .

5.2 Corrective Controller Existence for a Machine

(5.2.1) LEMMA. Let $\Sigma_{|s} = (A, Y, X, s, h)$ be the stable state machine induced by a machine $\Sigma = (A, Y, X, f, h)$, and let $\Sigma' = (A, Y, X, s', h)$ be a stable state machine. Let $\#X = n$. If the skeleton matrix $S^{(n-1)}(\Sigma)$ of the machine Σ is identical to the skeleton matrix $S^{(n-1)}(\Sigma')$ of the machine Σ' , i.e. $S^{(n-1)}(\Sigma) = S^{(n-1)}(\Sigma')$ then there is a controller C such that the stable state closed loop system $\Sigma_{c|s}$ is equivalent to the machine Σ' .

PROOF. Denote the states of Σ and Σ' as $X = \{x_1, x_2, \dots, x_n\}$. Recall that $\Sigma_{c|s}$ is equivalent to Σ' whenever $\gamma((x, \alpha), u) = (s'(x, u), \alpha')$ for all valid pairs $(x, u) \in X \times A$, γ is the stable transition function of $\Sigma_{c|s}$, and α and α' are states of the controller C . The controller C must ensure that for all valid pairs $(x_i, u) \in X \times A$ that whenever $s'(x_i, u) = x_k$ that $\gamma((x_i, \alpha), u) = (x_k, \alpha')$. Also recall that whenever $s'(x_i, u) = x_k$ the entry $S_{ik}^{(n-1)}(\Sigma')$ is a one and thus the entry $S_{ik}^{(n-1)}(\Sigma)$ is also a one, i.e. whenever $s'(x_i, u) = x_k$ the state x_k is stably reachable from the state x_i in Σ since $S^{(n-1)}(\Sigma) = S^{(n-1)}(\Sigma')$. Now for all stable transitions $s'(x_i, u) = x_k$, either $s(x_i, u) = x_k$ or by Theorem (5.1.3) a controller C can be constructed so that $\gamma((x_i, \alpha), u) =$

(x_k, α') , since x_k is stably reachable from x_i in Σ . Thus $\gamma((x, \alpha), u) = (s'(x, u), \alpha')$ for all valid pairs $(x, u) \in X \times A$, and the stable state closed loop system $\Sigma_{c|s}$ is equivalent to Σ' . ♦

Consider two $n \times n$ skeleton matrices S and S' . Let i and j be two integers. Define the relation $\geq$ for two skeleton matrices by $S \geq S'$ whenever $S_{ij} \geq S'_{ij}$ for all $i = 1, \dots, n$ and $j = 1, \dots, n$. Remember that all of the elements of S and S' are ones and zeros, so this relation will only fail when for some i and j , $S_{ij} = 0$ and $S'_{ij} = 1$.

(5.2.2) LEMMA. Let $\Sigma_{|s} = (A, Y, X, s, h)$ be the stable state machine induced by a machine $\Sigma = (A, Y, X, f, h)$, and let $\Sigma' = (A, Y, X, s', h)$ be a stable state machine. Let $\#X = n$. The following two statements are equivalent.

- (i) There is a controller C such that the stable state closed loop system $\Sigma_{c|s}$ is equivalent to the machine Σ' .
- (ii) The skeleton matrix $S^{(n-1)}(\Sigma)$ of the machine Σ satisfies $S^{(n-1)}(\Sigma) \geq S^{(n-1)}(\Sigma')$.

PROOF. Denote the states of Σ and Σ' as $X = \{x_1, x_2, \dots, x_n\}$. Recall that $\Sigma_{c|s}$ is equivalent to Σ' whenever $\gamma((x, \alpha), u) = (s'(x, u), \alpha')$ for all valid pairs $(x, u) \in X \times A$, where γ is the stable transition function of $\Sigma_{c|s}$, and α, α' are states of the controller C . The controller C must ensure that for all valid pairs $(x_i, u) \in X \times A$ that whenever $s'(x_i, u) = x_k$ that $\gamma((x_i, \alpha), u) = (x_k, \alpha')$. Also recall that whenever $s'(x_i, u) = x_k$ the entry $S_{ik}^{(n-1)}(\Sigma')$ is a one and thus the entry $S_{ik}^{(n-1)}(\Sigma)$ is also a one, i.e. whenever $s'(x_i, u) = x_k$ the state x_k is stably reachable from the state x_i in Σ since $S^{(n-1)}(\Sigma) \geq S^{(n-1)}(\Sigma')$. Now for all stable transitions $s'(x_i, u) = x_k$, either $s(x_i, u) = x_k$ or by Theorem (5.1.3) a controller C can be constructed so that $\gamma((x_i, \alpha), u) =$

(x_k, α') since x_k is stably reachable from x_i in Σ . Thus $\gamma((x, \alpha), u) = (s'(x, u), \alpha')$ for all valid pairs $(x, u) \in X \times A$, and the system $\Sigma_{c|s}$ is equivalent to Σ' .

Now assume there is a controller C such that the closed loop system $\Sigma_{c|s}$ is equivalent to Σ' and that $S^{(n-1)}(\Sigma) \not\geq S^{(n-1)}(\Sigma')$. There is then, at least one entry of $S^{(n-1)}(\Sigma')$ which is a one and the same entry of $S^{(n-1)}(\Sigma)$ is a zero. Suppose that the i, j entry is of this type. There is then a path from x_i to x_j in Σ' but not in Σ . This implies that for some point along this path that there is a stable transition $s'(x, u)$ which takes on a different state value than $s(x, u)$ and that this state is not stably reachable from x in Σ . Otherwise, x_j is stably reachable from x_i in Σ and the i, j entry of $S^{(n-1)}(\Sigma)$ is a one. Recall that $\Sigma_{c|s}$ is equivalent to Σ' whenever $\gamma((x', \alpha), u) = (s'(x', u), \alpha')$ for all valid pairs $(x', u) \in X \times A$. However, there is at least one pair, (x, u) , for which $s'(x, u)$ takes on a value that $\gamma((x, \alpha), u)$ cannot take on. Thus $\Sigma_{c|s}$ is not equivalent to Σ' contradicting our assumption. Thus whenever a controller C exists such that $\Sigma_{c|s}$ is equivalent to Σ' then the relation $S^{(n-1)}(\Sigma) \geq S^{(n-1)}(\Sigma')$ must hold. ♦

Whenever a stable state machine $\Sigma|_s = (A, Y, X, s, h)$ is strongly connected, the relation $S^{(n-1)}(\Sigma) \geq S^{(n-1)}(\Sigma')$ will hold for any $\Sigma' = (A, Y, X, s', h)$.

5.3 Corrective Controller Existence for a Family of Machines

(5.3.1) PROPOSITION. Let $M = \{\Sigma_1, \Sigma_2, \dots, \Sigma_q\}$ be a critical race family of q machines induced by a machine $\Sigma = (A, Y, X, f, h)$. Let $\Sigma' = (A, Y, X, s', h)$ be a

stable state machine and let $\#X = n$. Pick any machine Σ_i from M , $i = 1, \dots, q$. The following two statements are equivalent.

- (i) There exists a controller C_i such that the stable system $\Sigma_{ic|s}$ is equivalent to the machine Σ' .
- (ii) The skeleton matrix $S^{(n-1)}(\Sigma_i)$ satisfies $S^{(n-1)}(\Sigma_i) \geq S^{(n-1)}(\Sigma')$.

The previous given methods of controller design can be used to design a corrective controller for any one member of a critical race family of machines. The controller design to achieve this result is identical to the controller designed in the proof of Theorem (5.1.3). We now extend this result to designing a single controller C which acts as a corrective controller for all the members of the critical race family of machines.

Denote the connection of a corrective controller C with a member Σ_i of a race family of machines $M = \{\Sigma_1, \Sigma_2, \dots, \Sigma_q\}$ by Σ_{ic} . Also denote the family of such systems as $M_c = \{\Sigma_{1c}, \Sigma_{2c}, \dots, \Sigma_{qc}\}$, and the stable state family of systems as $M_{c|s} = \{\Sigma_{1c|s}, \Sigma_{2c|s}, \dots, \Sigma_{qc|s}\}$.

(5.3.2) THEOREM. Let $M = \{\Sigma_1, \Sigma_2, \dots, \Sigma_q\}$ be a critical race family of machines induced by a machine $\Sigma = (A, Y, X, f, h)$ that has a single critical race. Let $\Sigma' = (A, Y, X, s', h)$ be a stable state machine, and let $\#X = n$. The following two statements are equivalent.

- (i) There is a controller C such that all members of $M_{c|s}$ are equivalent to Σ' .
- (ii) The skeleton matrix $\mathcal{S}(M)$ satisfies $\mathcal{S}(M) \geq S^{(n-1)}(\Sigma')$.

PROOF. Recall a critical race family of machines $M = \{\Sigma_1, \dots, \Sigma_q\}$ is q deterministic stable state machines, all of which have the same input set A , the same output set Y , the same state set X , and the same output

function h . Let the critical race of the machine Σ be at the pair (x, u) and let the transition $f(x, u)$ have q outcomes denoted by $\{z_1, \dots, z_q\}$. The transition function f_i of the machine Σ_i assumes the value z_i for the pair (x, u) , $i = 1, \dots, q$, and is otherwise equal to the transition function f of Σ . All members of the critical race family of machines M differ only in their state transition function, and then only in the value of the transition function at one pair, that of the critical race (x, u) .

The critical race (x, u) in Σ induces up to $\#X$ critical races in the stable state machine $\Sigma|_s = (A, Y, X, s, h)$ at the pairs (X', u) , $X' \subset X$.

Recall that $\Sigma_{ic|s}$ is equivalent to Σ' whenever $\gamma((x, \alpha), u) = (s'(x, u), \alpha')$ for all valid pairs $(x, u) \in X \times A$, γ is the stable transition function of $\Sigma_{ic|s}$ and α, α' are states of the controller C . The controller C must ensure then that for all valid pairs $(x, u) \in X \times A$ that whenever $s'(x, u) = x'$ that $\gamma((x, \alpha), u) = (x', \alpha')$. Also recall that whenever $s'(x, u) = x'$ the entry of $S^{(n-1)}(\Sigma')$ indexed by the states x and x' is a one and thus the entry of $\mathcal{A}(M)$ indexed by x and x' .

Now pick a valid pair $(x, u) \in X \times A$. Then either $s_i(x, u)$ takes on a single value for all $i = 1, \dots, q$, or there are at least two $s_i(x, u)$, $i = 1, \dots, q$, which take on different values. The second case corresponds to a critical race at the pair (x, u) .

Now consider a path from a state x to a state x' which is stably reachable from x in M , there are two possibilities. There is an input list $w \in A^+$ such that $s_i(x, w) = x'$, $i = 1, \dots, q$, or there are input lists $w_i \in A^+$ such that $s_i(x, w_i) = x'$, $i = 1, \dots, q$.

Consider for our model matching problem that there are p transitions (x_j, u_j) , $j = 1, \dots, p$, for which $s'(x_j, u_j) \neq s_i(x_j, u_j)$ for all $i = 1, \dots, q$. Denote the next stable state of the stable transition $s'(x_j, u_j)$ as x_j' .

Let $\mathcal{L}$ of the p transitions to be changed have no critical race in the desired path, and let $m = p - \mathcal{L}$ transitions have a critical race in the desired path. We construct a controller $C = (A \times Y, A, \Xi, \phi, \eta)$ as follows.

We define the state transition function from the state α_0 by setting $\phi(\alpha_0, (z, u)) = \alpha_0$ for all $(z, u) \in X \times A \setminus (x_j, U_j)$ where $U_j \subset A$ is the set of input values which form stable combinations with the state x_j , and $\phi(\alpha_0, (x_j, U_j)) = \alpha_1, j = 1, \dots, p$. The value of the output function η for the same state is $\eta(\alpha_0, (z, u)) = u$ for all $(z, u) \in X \times A$.

The transition function from the state α_1 is defined by setting $\phi(\alpha_1, (x_j, u_j)) = \alpha_{j1}, \phi(\alpha_1, (x_j, U_j)) = \alpha_1, j = 1, \dots, p$, and $\phi(\alpha_1, (z, u)) = \alpha_0$ for all other $(z, u) \in X \times A$. The value of the output function at the same state is $\eta(\alpha_1, (z, u)) = u_j$ where $u_j \in U_j$.

Now first consider the $\mathcal{L}$ transitions which do not encounter a critical race. The rest of the controller construction for these transitions is identical to the controller constructed in (5.1.3), i.e. By the choice of x_j' there is a $w_j \in A^+, w_j = v_{j0}v_{j1}\dots v_{j(m(j)-1)}$ such that $s(x_j, w_j) = x_j'$. Denote $x_{j1} := s(x_j, v_{j0}), x_{j2} := s(x_{j1}, v_{j1}), x_{j3} := s(x_{j2}, v_{j2}), \dots, x_{j(m(j)-1)} := s(x_{j(m(j)-2)}, v_{j(m(j)-2)}), x_j' := s(x_{j(m(j)-1)}, v_{j(m(j)-1)}), j = 1, \dots, \mathcal{L}$.

And the rest of the transition function ϕ and output function η are given by for $j = 1, \dots, \mathcal{L}$ by

$$\phi(\alpha_{j1}, (z, u)) = \alpha_{j1} \text{ for all } (z, u) \in X \times A \setminus (x_{j1}, u_j) \text{ and } \phi(\alpha_{j1}, (x_{j1}, u_j)) = \alpha_{j2}$$

$$\eta(\alpha_{j1}, (z, u)) = v_{j0} \text{ for all } (z, u) \in X \times A$$

$$\phi(\alpha_{j2}, (z, u)) = \alpha_{j2} \text{ for all } (z, u) \in X \times A \setminus (x_{j2}, u_j) \text{ and } \phi(\alpha_{j2}, (x_{j2}, u_j)) = \alpha_{j3}$$

$$\eta(\alpha_{j2}, (z, u)) = v_{j1} \text{ for all } (z, u) \in X \times A$$

for $3 < k < m(j)-1$

$$\phi(\alpha_{jk}, (z, u)) = \alpha_{jk} \text{ for all } (z, u) \in X \times A \setminus (x_{jk}, u_j) \text{ and } \phi(\alpha_{jk}, (x_{jk}, u_j)) = \alpha_{jk+1}$$

$$\eta(\alpha_{jk}, (z, u)) = v_{jk-1} \text{ for all } (z, u) \in X \times A$$

and for $k = m(j)$

$$\phi(\alpha_{jm(j)}, (z, u)) = \alpha_{jm(j)} \text{ for all } (z, u) \in X \times A \setminus (x_j', A - u_j) \text{ and } \phi(\alpha_{jm(j)}, (x_j', A - u_j)) = \alpha_0$$

$$\eta(\alpha_{jm(j)}, (z, u)) = v_{j(m(j)-1)} \text{ for all } (z, u) \in X \times A$$

Now consider the m transitions that encounter a critical race in the desired path. From the state α_1 when the pair to change is recognized, a set of controller states and transitions as in the controller of (5.1.1) will take the machine from x_j to the critical race. The critical race is then encountered and the machine goes to one of its next stable states. The controller needs a state like the state α_1 to wait in until the next stable state of the critical race is reached. The controller then needs another set of states and transitions like (5.1.1) to send each outcome of the race to the state x_j' , i.e. the part of the controller for the transitions that encounter a race will be of the form $C((x_j, u_j), w_j, z_j')$, then a wait state for the outcome of the race, then $C((z_k', u_k'), w_k, x_j')$, $k = 1, \dots, q(j)$, $j = \ell+1, \dots, p$. Here the z_k' 's are the possible outcomes of the race, and z_j' is the race in the j th transition path. The controller part for the transitions that encounter the critical race, branch into $q(j)$ parallel parts to fix the effect of the race and then converge back at the terminal state of the controller.

Now assume that there is a controller C such that all members of $M_{c|s}$ are equivalent to Σ' and that $\mathcal{N}(M) \not\geq S^{(n-1)}(\Sigma')$. This implies that there is at least one state x' which is stably reachable from a state x in Σ' , but not in all the machines Σ_i of M , $i = 1, \dots, q$. There is then at least one Σ_i for which x' is not stably reachable from x . There is then a path from x to x' in Σ' and not in at least one Σ_i . This implies that for some point along this path that there is a stable transition $s'(x, u)$ which takes on a different state value than $s_i(x, u)$ and that this state is not stably reachable from x

in Σ . Recall that when $\Sigma_{ic|s}$ is equivalent to Σ' , the stable transition function of $\Sigma_{ic|s}$ must take on the same value as the transition function of Σ' at all valid pairs. We have shown in Theorem (5.2.2), that this is not possible when $S^{(n-1)}(\Sigma_i) \not\geq S^{(n-1)}(\Sigma')$. Thus there is at least one $\Sigma_{ic|s}$ which is not equivalent to Σ' contradicting our assumption. Thus whenever a controller C exists such that all members of $M_{c|s}$ are equivalent to Σ' then the relation $\mathcal{S}(M) \geq S^{(n-1)}(\Sigma')$ must hold. ♦

In effect, Theorem (5.3.2) gives a controller C that fixes the effects of a critical race. The corrective controller ensures all members Σ_i of the critical race family of machines M exhibit the same deterministic behavior. We must ensure for consistency that the corrective controller does not need to output different values to two or more different members of the critical race family when they are in the same state receiving the same reference input.

(5.3.3) PROPOSITION. Consider a critical race family of machines $M = \{\Sigma_1, \Sigma_2, \dots, \Sigma_q\}$ induced by a machines Σ with a single critical race. Let C be a controller such that all members of $M_{c|s}$ are equivalent to a machine Σ' . Pick any valid pair (x,v) from Σ . The controller C outputs the same value to all members Σ_i of M , $i = 1, \dots, q$, at the state x with the reference input v .

PROOF. Recall that the members of a critical race family of machines all have the same input set, the same output set, the same state set, and the same output function. Their transition functions are also the same at all valid pairs except that of the critical race.

Consider now a transition from a potentially stable state x of M to a potentially stable state x' stably reachable in the family M from the state x . By the choice of x' there is either

- (i) A path from x to x' which is the same in all members of M , or
- (ii) A set of machines $M_f \subset M$, which all have the same path from x to x' , and a set of machines $M_r = M - M_f$ each having a different path from x to x' . This difference is due to the critical race.

Consider case (i). The path from x to x' is the identical in all members of the family M . Thus, since M is a critical race family of machines and the members of M have the same transition functions except at the critical race, the same input list will drive all the members of M from x to x' . We can see this by looking at the controller constructed in (5.3.2), that for any transitions that do not encounter a critical race, the same input list w is provided to all the machines Σ_i of M , $i = 1, \dots, q$. Thus the corrective controller C provides the same value to each member of M for the same state and reference input.

Now consider a transition that encounters a critical race, case (ii). An identical input list is provided to all machines Σ_i of M , $i = 1, \dots, q$, until the critical race is encountered. Only after the critical race, are different inputs provided depending on the machine Σ_i of M , $i = 1, \dots, q$. However, these different output values are provided at different values of the states, since the members Σ_i of M , $i = 1, \dots, q$, differ in their next stable state after a critical race. Thus the same value is provided by the controller C to all machines Σ_i of M , $i = 1, \dots, q$, at the same state and reference input. ♦

The results obtained in this paper are valid for any family of machines which has the same properties as a critical race family of machines.

CHAPTER 6 EXAMPLES

6.1 Example 1

Consider the stable state machine $\Sigma|_s = (A, Y, X, s, h)$ induced by a machine $\Sigma = (A, Y, X, f, h)$. Denote the state set of Σ by $X = \{x_0, x_1, x_2\}$, the input set of Σ by $A = \{a, b, c\}$, and the output set of Σ by $Y = X$. The transition function f of Σ is defined by

$$f(x_0, a) = x_0 \quad f(x_1, a) = x_2 \quad f(x_2, a) = x_2$$

$$f(x_0, b) = x_1 \quad f(x_1, b) = x_1 \quad f(x_2, b) = x_1$$

$$f(x_0, c) = \text{critical race} \quad f(x_1, c) = x_1 \quad f(x_2, c) = x_2$$

The possible outcomes of the critical race (x_0, c) are the states x_1 and x_2 . The output function h of Σ is defined by $h(x_i) = x_i$ for all $x_i \in X$. The faulty machine Σ induces a critical race family M of two machines $M = \{\Sigma_1, \Sigma_2\}$, where $\Sigma_1 = (A, Y, X, f_1, h)$ and $\Sigma_2 = (A, Y, X, f_2, h)$. The transition function f_1 of Σ_1 is defined by $f_1(x_i, u) = f(x_i, u)$ for all $(x_i, u) \in X \times A \setminus (x_0, c)$, and $f_1(x_0, c) = x_1$. The transition function f_2 of Σ_2 is defined by $f_2(x_i, u) = f(x_i, u)$ for all $(x_i, u) \in X \times A \setminus (x_0, c)$, and $f_2(x_0, c) = x_2$.

The machines Σ_1 and Σ_2 induce the stable state machines $\Sigma_1|_s$ and $\Sigma_2|_s$ respectively. The stable transition functions s_1 of Σ_1 and s_2 of Σ_2 are defined by

$$s_1(x_0, a) = x_0 \quad s_1(x_1, a) = x_2 \quad s_1(x_2, a) = x_2$$

$$s_1(x_0, b) = x_1 \quad s_1(x_1, b) = x_1 \quad s_1(x_2, b) = x_1$$

$$s_1(x_0, c) = x_1 \quad s_1(x_1, c) = x_1 \quad s_1(x_2, c) = x_2$$

and

$$s_2(x_0, a) = x_0 \quad s_2(x_1, a) = x_2 \quad s_2(x_2, a) = x_2$$

$$s_2(x_0, b) = x_1 \quad s_2(x_1, b) = x_1 \quad s_2(x_2, b) = x_1$$

$$s_2(x_0, c) = x_2 \quad s_2(x_1, c) = x_1 \quad s_2(x_2, c) = x_2$$

The one-step matrices of stable transitions $R(\Sigma_1)$ and $R(\Sigma_2)$ of Σ_1 and Σ_2 respectively are

$$R(\Sigma_1) = \begin{pmatrix} \{a\} & \{b, c\} & N \\ N & \{b, c\} & \{a\} \\ N & \{b\} & \{a, c\} \end{pmatrix} \quad R(\Sigma_2) = \begin{pmatrix} \{a\} & \{b\} & \{c\} \\ N & \{b, c\} & \{a\} \\ N & \{b\} & \{a, c\} \end{pmatrix}$$

The matrices of stable transitions $R^{(2)}(\Sigma_1)$ and $R^{(2)}(\Sigma_2)$ of Σ_1 and Σ_2 respectively are

$$R^{(2)}(\Sigma_1) = \begin{pmatrix} \{a, aa\} & \{b, c, ab, ac, bb, bc, cb, cc\} & \{ba, ca\} \\ N & \{b, c, bb, bc, cb, cc, ab\} & \{a, ba, ca, aa, ac\} \\ N & \{b, bb, bc, ab, cb\} & \{a, c, ba, aa, ac, ca, cc\} \end{pmatrix}$$

$$R^{(2)}(\Sigma_2) = \begin{pmatrix} \{a, aa\} & \{b, ab, bb, bc, cb\} & \{c, ac, ba, ca, cc\} \\ N & \{b, c, bb, bc, cb, cc, ab\} & \{a, ba, ca, aa, ac\} \\ N & \{b, bb, bc, ab, cb\} & \{a, c, ba, aa, ac, ca, cc\} \end{pmatrix}$$

The one-step skeleton matrices $S(\Sigma_1)$ and $S(\Sigma_2)$ of Σ_1 and Σ_2 respectively are

$$S(\Sigma_1) = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \quad S(\Sigma_2) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

The skeleton matrices $S^{(2)}(\Sigma_1)$ and $S^{(2)}(\Sigma_2)$ of Σ_1 and Σ_2 respectively are

$$S^{(2)}(\Sigma_1) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \quad S^{(2)}(\Sigma_2) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

The skeleton matrix $\mathcal{H}(M)$ of M is

$$\mathcal{H}(M) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

We choose the machine $\Sigma' = (A, Y, X, s', h)$ as the desired model to match. The transition function s' of Σ' is defined by

$$s'(x_0, a) = x_0 \quad s'(x_1, a) = x_2 \quad s'(x_2, a) = x_2$$

$$s'(x_0, b) = x_1 \quad s'(x_1, b) = x_1 \quad s'(x_2, b) = x_1$$

$$s'(x_0, c) = x_2 \quad s'(x_1, c) = x_1 \quad s'(x_2, c) = x_2$$

The one-step matrix of stable transitions $R(\Sigma')$ of Σ' is

$$R(\Sigma') = \begin{pmatrix} \{a\} & \{b\} & \{c\} \\ N & \{b, c\} & \{a\} \\ N & \{b\} & \{a, c\} \end{pmatrix}$$

The matrix of stable transitions $R^{(2)}(\Sigma')$ of Σ' is

$$R^{(2)}(\Sigma') = \begin{pmatrix} \{a, aa\} & \{b, ab, bb, bc, cb\} & \{c, ac, ba, ca, cc\} \\ N & \{b, c, bb, bc, cb, cc, ab\} & \{a, ba, ca, aa, ac\} \\ N & \{b, bb, bc, ab, cb\} & \{a, c, ba, aa, ac, ca, cc\} \end{pmatrix}$$

The one-step skeleton matrix $S(\Sigma')$ of Σ' is

$$S(\Sigma') = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

The skeleton matrix $S^{(2)}(\Sigma')$ of Σ' is

$$S^{(2)}(\Sigma') = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

On comparison of $\Sigma|_s$ and Σ' , we can see that $s = s'$ at all pairs except (x_0, c) . The corrective controller must ensure that whenever the pair (x_0, c) occurs after a stable combination of the state x_0 that the closed loop system $\Sigma_{c|s}$ enters the state x_2 as its next stable state. Then the closed loop system $\Sigma_{c|s}$ will be equivalent to the model Σ' .

We construct the corrective controller $C = (A \times Y, A, \Xi, \phi, \eta)$. The state set Ξ of the controller consists of 4 states, which we denote by the symbols $\Xi = \{\alpha_0, \alpha_1, \alpha_2, \alpha_3\}$. We define the transition function from the state α_0 by setting $\phi(\alpha_0, (z, u)) = \alpha_0$ for all $(z, u) \in X \times A \setminus (x_0, a)$ and $\phi(\alpha_0, (x_0, a)) = \alpha_1$. The value of the output function for the same state is $\eta(\alpha_0, (z, u)) = u$ for all $(z, u) \in X \times A$.

The transition function from the state α_1 is defined by setting $\phi(\alpha_1, (x_0, c)) = \alpha_2$, $\phi(\alpha_1, (x_0, a)) = \alpha_1$ and $\phi(\alpha_1, (z, u)) = \alpha_0$ for all other $(z, u) \in X \times A$. The value of the output function at the same state is $\eta(\alpha_1, (z, u)) = a$ for all $(z, u) \in X \times A$.

The rest of the transition function and output function are given by $\phi(\alpha_2, (z, u)) = \alpha_2$ for all $(z, u) \in X \times A \setminus (x_1, c)$ and $\phi(\alpha_2, (x_1, c)) = \alpha_3$
 $\eta(\alpha_2, (z, u)) = b$ for all $(z, u) \in X \times A$
 $\phi(\alpha_3, (z, u)) = \alpha_3$ for all $(z, u) \in X \times A \setminus (x_2, \{a, b\})$ and $\phi(\alpha_3, (x_2, \{a, b\})) = \alpha_0$
 $\eta(\alpha_3, (z, u)) = a$ for all $(z, u) \in X \times A$ ♦

6.2 Example 2

Consider the machine Σ described by the following table

Table 6.2.1. The machine Σ of Example 6.2

x_1x_2	X_1X_2			
	u_1u_2			
	00	01	11	10
00	11	00	10	01
01	11	00	11	01
11	11	00	10	11
10	11	10	10	11

We can see the machine Σ has two races at the state/input combinations $x_1x_2/u_1u_2 = 00/00$ and $11/01$. The race $x_1x_2/u_1u_2 = 00/00$, is a non-critical race, the next stable state of the state/input pair $00/00$ is always 11. The race $x_1x_2/u_1u_2 = 11/01$, however, is a critical race. The state/input pair $11/01$ has three possible next stable states, 01, 11, and 10.

We denote the machine Σ by $\Sigma = (A, Y, X, f, h)$, where the state set of Σ is the characters $X = \{x_0, x_1, x_2, x_3\}$, and $x_0 = 00$, $x_1 = 01$, $x_2 = 11$, and $x_3 = 10$. The input set of Σ is denoted by $A = \{a, b, c, d\}$, where $a = 00$, $b = 01$, $c = 11$, and $d = 10$.

We can define the transition function f of Σ by

$$f(x_0, a) = x_2 \quad f(x_1, a) = x_2 \quad f(x_2, a) = x_2 \quad f(x_3, a) = x_2$$

$$f(x_0, b) = x_0 \quad f(x_1, b) = x_0 \quad f(x_2, b) = \text{critical race} \quad f(x_3, b) = x_3$$

$$f(x_0, c) = x_3 \quad f(x_1, c) = x_2 \quad f(x_2, c) = x_3 \quad f(x_3, c) = x_3$$

$$f(x_0, d) = x_1 \quad f(x_1, d) = x_1 \quad f(x_2, d) = x_2 \quad f(x_3, d) = x_2$$

And we define the output function h of Σ by $h(x_i) = x_i$ for all $x_i \in X$. The possible outcomes of the critical race are the states x_0 , x_1 , and x_3 .

The faulty machine Σ induces a critical race family M of three machines, $M = \{\Sigma_1, \Sigma_2, \Sigma_3\}$, where $\Sigma_1 = (A, Y, X, f_1, h)$, $\Sigma_2 = (A, Y, X, f_2, h)$, and $\Sigma_3 = (A, Y, X, f_3, h)$. The transition function f_1 of Σ_1 is defined by $f_1(x_i, u) = f(x_i, u)$ for all $(x_i, u) \in X \times A \setminus (x_2, b)$, and $f_1(x_2, b) = x_0$. The transition function f_2 of Σ_2 is defined by $f_2(x_i, u) = f(x_i, u)$ for all $(x_i, u) \in X \times A \setminus (x_2, b)$, and $f_2(x_2, b) = x_1$. The transition function f_3 of Σ_3 is defined by $f_3(x_i, u) = f(x_i, u)$ for all $(x_i, u) \in X \times A \setminus (x_2, b)$, and $f_3(x_2, b) = x_3$.

The machines Σ_1 , Σ_2 , and Σ_3 induce the stable state machines $\Sigma_{1|s}$, $\Sigma_{2|s}$, and $\Sigma_{3|s}$ respectively. The stable transition functions s_1 of Σ_1 , s_2 of Σ_2 , and s_3 of Σ_3 are defined by

$$\begin{aligned} s_1(x_0, a) &= x_2 & s_1(x_1, a) &= x_2 & s_1(x_2, a) &= x_2 & s_1(x_3, a) &= x_2 \\ s_1(x_0, b) &= x_0 & s_1(x_1, b) &= x_0 & s_1(x_2, b) &= x_0 & s_1(x_3, b) &= x_3 \\ s_1(x_0, c) &= x_3 & s_1(x_1, c) &= x_3 & s_1(x_2, c) &= x_3 & s_1(x_3, c) &= x_3 \\ s_1(x_0, d) &= x_1 & s_1(x_1, d) &= x_1 & s_1(x_2, d) &= x_2 & s_1(x_3, d) &= x_2 \end{aligned}$$

$$\begin{aligned} s_2(x_0, a) &= x_2 & s_2(x_1, a) &= x_2 & s_2(x_2, a) &= x_2 & s_2(x_3, a) &= x_2 \\ s_2(x_0, b) &= x_0 & s_2(x_1, b) &= x_0 & s_2(x_2, b) &= x_0 & s_2(x_3, b) &= x_3 \\ s_2(x_0, c) &= x_3 & s_2(x_1, c) &= x_3 & s_2(x_2, c) &= x_3 & s_2(x_3, c) &= x_3 \\ s_2(x_0, d) &= x_1 & s_2(x_1, d) &= x_1 & s_2(x_2, d) &= x_2 & s_2(x_3, d) &= x_2 \end{aligned}$$

and

$$\begin{aligned} s_3(x_0, a) &= x_2 & s_3(x_1, a) &= x_2 & s_3(x_2, a) &= x_2 & s_3(x_3, a) &= x_2 \\ s_3(x_0, b) &= x_0 & s_3(x_1, b) &= x_0 & s_3(x_2, b) &= x_3 & s_3(x_3, b) &= x_3 \\ s_3(x_0, c) &= x_3 & s_3(x_1, c) &= x_3 & s_3(x_2, c) &= x_3 & s_3(x_3, c) &= x_3 \\ s_3(x_0, d) &= x_1 & s_3(x_1, d) &= x_1 & s_3(x_2, d) &= x_2 & s_3(x_3, d) &= x_2 \end{aligned}$$

We can see from the stable transition functions s_1 of Σ_1 and s_2 of Σ_2 that the stable state machines $\Sigma_{1|s}$ and $\Sigma_{2|s}$ are identical.

The one-step matrices of stable transitions $R(\Sigma_1)$, $R(\Sigma_2)$ and $R(\Sigma_3)$ of Σ_1 , Σ_2 , and Σ_3 respectively are

$$R(\Sigma_1) = R(\Sigma_2) = \begin{pmatrix} \{b\} & \{d\} & \{a\} & \{c\} \\ \{b\} & \{d\} & \{a\} & \{c\} \\ \{b\} & N & \{a,d\} & \{c\} \\ N & N & \{a,d\} & \{b,c\} \end{pmatrix} \quad R(\Sigma_3) = \begin{pmatrix} \{b\} & \{d\} & \{a\} & \{c\} \\ \{b\} & \{d\} & \{a\} & \{c\} \\ N & N & \{a,d\} & \{b,c\} \\ N & N & \{a,d\} & \{b,c\} \end{pmatrix}$$

The one-step skeleton matrices $S(\Sigma_1)$, $S(\Sigma_2)$, and $S(\Sigma_3)$ of Σ_1 , Σ_2 , and Σ_3 respectively are

$$S(\Sigma_1) = S(\Sigma_2) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \quad S(\Sigma_3) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

The skeleton matrices $S^{(3)}(\Sigma_1)$, $S^{(3)}(\Sigma_2)$, and $S^{(3)}(\Sigma_3)$ of Σ_1 , Σ_2 , and Σ_3 respectively are

$$S^{(3)}(\Sigma_1) = S^{(3)}(\Sigma_2) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad S^{(3)}(\Sigma_3) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

The skeleton matrix $\mathcal{S}(M)$ of M is

$$\mathcal{S}(M) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

We first choose the machine $\Sigma' = (A, Y, X, s', h)$ as the desired model to match. We define the transition function s' of Σ' by

$$\begin{aligned} s'(x_0, a) &= x_2 & s'(x_1, a) &= x_2 & s'(x_2, a) &= x_2 & s'(x_3, a) &= x_2 \\ s'(x_0, b) &= x_0 & s'(x_1, b) &= x_0 & s'(x_2, b) &= x_0 & s'(x_3, b) &= x_3 \end{aligned}$$

$$\begin{aligned}
s'(x_0, c) &= x_3 & s'(x_1, c) &= x_3 & s'(x_2, c) &= x_3 & s'(x_3, c) &= x_3 \\
s'(x_0, d) &= x_1 & s'(x_1, d) &= x_1 & s'(x_2, d) &= x_2 & s'(x_3, d) &= x_2
\end{aligned}$$

Which is how Σ would behave if it were a race-free machine. We can see that the machine Σ' is identical to the machines $\Sigma_1|_s$ and $\Sigma_2|_s$. Thus the skeleton matrix $S^{(3)}(\Sigma')$ of Σ' is

$$S^{(3)}(\Sigma') = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

Recall that the skeleton matrix of the critical race family $\mathcal{M}$ is

$$\mathcal{A}(\mathcal{M}) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Comparing $\mathcal{A}(\mathcal{M})$ to $S^{(3)}(\Sigma')$ we find that $\mathcal{A}(\mathcal{M}) \not\geq S^{(3)}(\Sigma')$, so by Theorem (5.3.2) there is no controller C such that all members of $\mathcal{M}_{c|s}$ are equivalent to Σ' . Specifically, stable transitions from the states x_3 and x_2 to the states x_0 and x_1 are not achievable by $\Sigma_3|_s$, so there is no controller C such that the machine $\Sigma_{3c|s}$ is equivalent to Σ' .

We next choose the machine $\Sigma'' = (A, Y, X, s'', h)$ as the desired model to match. We define the transition function s'' of Σ'' by

$$\begin{array}{llll}
s''(x_0, a) = x_2 & s''(x_1, a) = x_2 & s''(x_2, a) = x_2 & s''(x_3, a) = x_2 \\
s''(x_0, b) = x_0 & s''(x_1, b) = x_0 & s''(x_2, b) = x_3 & s''(x_3, b) = x_3 \\
s''(x_0, c) = x_3 & s''(x_1, c) = x_3 & s''(x_2, c) = x_3 & s''(x_3, c) = x_3 \\
s''(x_0, d) = x_1 & s''(x_1, d) = x_1 & s''(x_2, d) = x_2 & s''(x_3, d) = x_2
\end{array}$$

We can see that the machine Σ'' is identical to the machine $\Sigma_3|_S$.

Thus the skeleton matrix of Σ'' is

$$S^{(3)}(\Sigma'') = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

Comparing $\mathcal{M}$ with $S^{(3)}(\Sigma'')$ we find that $\mathcal{M} \geq S^{(3)}(\Sigma'')$. We construct a controller $C = (A \times Y, A, \Xi, \phi, \eta)$ so that all members of $M_{c|_S}$ are equivalent to Σ'' as follows.

The state set Ξ of the controller consists of 3 states, which we denote by the symbols $\Xi = \{\alpha_0, \alpha_1, \alpha_2\}$. We define the transition function ϕ of the controller C from the state α_0 by setting $\phi(\alpha_0, (z, u)) = \alpha_0$ for all $(z, u) \in X \times A \setminus \{(x_2, a), (x_2, d)\}$ and $\phi(\alpha_0, (x_2, a)) = \alpha_1$, $\phi(\alpha_0, (x_2, d)) = \alpha_1$. The value of the output function for the same state is $\eta(\alpha_0, (z, u)) = u$ for all $(z, u) \in X \times A$.

The transition function from the state α_1 is defined by setting $\phi(\alpha_1, (x_2, b)) = \alpha_2$, $\phi(\alpha_1, (x_2, a)) = \alpha_1$, $\phi(\alpha_1, (x_2, d)) = \alpha_1$, and $\phi(\alpha_1, (z, u)) = \alpha_0$ for all other $(z, u) \in X \times A$. The value of the output function at the same state is $\eta(\alpha_1, (z, u)) = d$ for all $(z, u) \in X \times A$.

The rest of the transition function and output function are given by $\phi(\alpha_2, (z, u)) = \alpha_2$ for all $(z, u) \in X \times A \setminus (x_3, \{a, c, d\})$ and $\phi(\alpha_2, (x_3, \{a, c, d\})) = \alpha_0$ $\eta(\alpha_2, (z, u)) = c$ for all $(z, u) \in X \times A$ ♦

CHAPTER 7 OTHER RESULTS

7.1 Multiple Races

The present work deals with machines with at most one critical race. This can be extended to cover the case of multiple critical races in a machine. The critical race family of machines will be slightly modified, and the corrective controller will be more complex, but the results on the existence of a corrective controller, Theorem (5.3.2), are valid with the skeleton matrix of the critical race family of machines induced by a machine with multiple critical races.

Consider such a machine $\Sigma = (A, Y, X, f, h)$ and let Σ have k critical races pairs $(x_i, u_i) \in X \times A, i = 1, \dots, k$. Denote the number of outcomes of the critical race (x_i, u_i) by $q(i), i = 1, \dots, k$. There are then $\prod_{i=1}^k q(i)$ permutations of critical race outcomes in Σ , we denote this number by $q := \prod_{i=1}^k q(i)$. We

build the critical race family of q machines $M = \{\Sigma_1, \dots, \Sigma_q\}$, all having the same input set, the same output set, the same state set, and the same output function, and whose transition functions are identical at all points except for those of the critical race pairs $(x_i, u_i), i = 1, \dots, k$. At these points the value of the transition function f_j of the machine Σ_j takes on the values corresponding to the j th permutation of the critical race outcomes, $j = 1, \dots, q$.

Consider, for example, a machine Σ with two critical races, one having two outcomes and the other having three outcomes. Denote the two critical races by (x_1, u_1) and (x_2, u_2) , and their possible outcomes by $\{x_{11}, x_{12}\}$ and $\{x_{21}, x_{22}, x_{23}\}$ respectively. There are six possible permutations of critical race outcomes, and we construct the critical race family of machines $M = \{\Sigma_1, \Sigma_2, \dots, \Sigma_6\}$ which has six members. The transition function f_1 of the machine Σ_1 at the critical race pairs is defined by $f_1(x_1, u_1) := x_{11}$ and $f_1(x_2, u_2) := x_{21}$. The rest of the transition functions f_j of Σ_j , $j = 2, \dots, 6$, are defined at the critical races by $f_2(x_1, u_1) := x_{11}$ and $f_2(x_2, u_2) := x_{22}$, $f_3(x_1, u_1) := x_{11}$ and $f_3(x_2, u_2) := x_{23}$, $f_4(x_1, u_1) := x_{12}$ and $f_4(x_2, u_2) := x_{21}$, $f_5(x_1, u_1) := x_{12}$ and $f_5(x_2, u_2) := x_{22}$, $f_6(x_1, u_1) := x_{12}$ and $f_6(x_2, u_2) := x_{23}$.

The skeleton matrix $\mathcal{S}(M)$ of a critical race family of machines M induced by a machine Σ with one or more critical races is constructed as in (5.4.2).

This leads to the following two results.

(7.1.1) PROPOSITION. Let $M = \{\Sigma_1, \Sigma_2, \dots, \Sigma_q\}$ be a critical race family of q machines induced by a machine $\Sigma = (A, Y, X, f, h)$. Let $\Sigma' = (A, Y, X, s', h)$ be a stable state machine and let $\#X = n$. Pick any machine Σ_i from M , $i = 1, \dots, q$. The following two statements are equivalent.

- (i) There exists a controller C_i such that the stable state closed loop system Σ_{icls} is equivalent to the machine Σ' .
- (ii) The skeleton matrix $S^{(n-1)}(\Sigma_i)$ satisfies $S^{(n-1)}(\Sigma_i) \geq S^{(n-1)}(\Sigma')$.

(7.1.2) THEOREM. Let $M = \{\Sigma_1, \Sigma_2, \dots, \Sigma_q\}$ be a critical race family of machines induced by a machine $\Sigma = (A, Y, X, f, h)$ that has one or more

critical races. Let $\Sigma' = (A, Y, X, s', h)$ be a stable state machine and let $\#X = n$. The following two statements are equivalent.

- (i) There is a controller C such that all members of $M_{c|s}$ are equivalent to Σ' .
- (ii) The skeleton matrix $\mathcal{A}(M)$ satisfies $\mathcal{A}(M) \geq S^{(n-1)}(\Sigma')$.

The proof for necessity is the same as that given for Theorem (5.3.2), however, the construction of the controller to show sufficiency is involved and not complete at this time.

7.2 Isomorphic Machines

In control theory, the outputs of systems are not usually relabeled. The outputs of physical systems correspond to specific physical things, like airplane pitch or roll, so relabeling the outputs does not make sense. However, the states of a physical system can be relabeled through a state transformation, and the inputs of a system can be relabeled through precompensation.

Currently our model matching problem involves finding a controller such that the closed loop system of (1.1) is equivalent to some desired model. In some cases two machines differ from each other only by the names of their states, the names of their outputs, the names of their inputs, but otherwise exhibit identical properties. Two such machines are usually said to be "isomorphic," we can also use this notion as a method of comparison between two machines.

Consider two machines $\Sigma = (A, Y, X, f, h)$ and $\Sigma' = (A, Y, X', f, h')$. A *morphism* or *state mapping* $\Psi : \Sigma \rightarrow \Sigma'$ is a partial function $\Psi : X \rightarrow X'$ satisfying

i) $\Psi[x_0] = x'_0$, i.e. the initial state of Σ is mapped to the initial state of Σ' .

ii) $f(\Psi[x], u) \subset \Psi[f(x, u)]$

iii) $h'(\Psi[x], u) \subset h(x, u)$

for all $x \in X$ and $u \in A$.

A state mapping Ψ is *proper* if $f(x, u) \neq \emptyset$ and $\Psi[x] \neq \emptyset$ imply $\Psi[f(x, u)] \neq \emptyset$ for all $x \in X$ and $u \in A$. From this we can deduce

$$f(\Psi[x], w) = \Psi[f(x, w)] \text{ if } \Psi[x] \neq \emptyset$$

$$h'(\Psi[x], w) = h(x, w) \text{ if } \Psi[x] \neq \emptyset.$$

The identity mapping $1_\Sigma : \Sigma \rightarrow \Sigma$, $1_\Sigma : X \rightarrow X$ is a state mapping.

The composition of two state mappings is also a state mapping, and if two state mappings are proper so is their composition.

Two machines $\Sigma = (A, Y, X, f, h)$ and $\Sigma' = (A', Y', X', f', h')$ are said to be *isomorphic* if there exists three bijective maps $\Psi_1 : X \rightarrow X'$, $\Psi_2 : A \rightarrow A'$, and $\Psi_3 : Y \rightarrow Y'$ such that, for all states $x \in X$ and all input characters $u \in A$, we have

$$\Psi_1[f(x, u)] = f'[\Psi_1(x), \Psi_2(u)], \text{ and}$$

$$\Psi_3[h(x, u)] = h'[\Psi_1(x), \Psi_2(u)].$$

The triple (Ψ_1, Ψ_2, Ψ_3) is referred to as an isomorphism between the machines Σ and Σ' . Intuitively, two machines are isomorphic whenever one can be obtained from the other simply by renaming the states, the input characters, and the output characters.

Using the notion of isomorphic machines we can restate the results of Theorem (5.1.3), Lemma (5.2.2), Proposition (5.3.1), and Theorem (5.3.2) as follows.

(7.2.1) PROPOSITION. Let $\Sigma_{|s} = (A, Y, X, s, h)$ be the stable state machine induced by a machine $\Sigma = (A, Y, X, f, h)$. Choose k states $x_i \in X, i = 1, \dots, k$, and k sets of input values $A_i \subset A$ where A_i is a set of inputs which form valid pairs with the state $x_i, i = 1, \dots, k$. Let $x_i' \in X$ be any state stably reachable from $x_i, i = 1, \dots, k$. Then, there is a controller C such that the stable state closed loop system $\Sigma_{c|s}$ is isomorphic to a machine Σ' that has the following properties:

- (i) $\Sigma' = (A, Y, X, s', h)$ has the same input, output, and state sets, as well as the same output function, as the machine $\Sigma_{|s}$.
- (ii) The transition function s' of Σ' is equal to the transition function s of $\Sigma_{|s}$ at all but the pairs (x_i, A_i) at which it takes the value $x_i', i = 1, \dots, k$. i.e. $s'(z, u) = s(z, u)$ for all $(z, u) \in X \times A \setminus (x_i, A_i)$ and $s'(x_i, A_i) = x_i', i = 1, \dots, k$.

The isomorphism $\Psi : \Sigma_{c|s} \rightarrow \Sigma'$ consists of three bijective maps $\Psi = \{\Psi_1, \Psi_2, \Psi_3\}$. Two of the maps $\Psi_2 : A \rightarrow A$ and $\Psi_3 : Y \rightarrow Y$ are identity maps. The state map $\Psi_1 : (X \times \alpha) \rightarrow X$ relabels the states of the stable state closed loop system $\Sigma_{c|s}$ to correspond with the states of Σ' , i.e. $\Psi_1((x, \alpha)) = x$ for all $x \in X$, here α is a state of the controller.

(7.2.2) PROPOSITION. Let $\Sigma_{|s} = (A, Y, X, s, h)$ be the stable state machine induced by a machine $\Sigma = (A, Y, X, f, h)$, and let $\Sigma' = (A, Y, X', s', h')$ be a stable state machine. Let $\#X = n$, and $\#X' = n$. If the skeleton matrix $S^{(n-1)}(\Sigma)$ of the machine Σ satisfies $S^{(n-1)}(\Sigma) \geq S^{(n-1)}(\Sigma')$, then there is a

controller C such that the stable state closed loop system $\Sigma_{c|s}$ is isomorphic to the machine Σ' .

Referring to (7.2.2), denote the states of Σ as $X = \{x_0, x_1, \dots, x_{n-1}\}$ and the states of Σ' as $X' = \{x'_0, x'_1, \dots, x'_{n-1}\}$. Since $S^{(n-1)}(\Sigma) \geq S^{(n-1)}(\Sigma')$ then the state x_j is stably reachable from the state x_i in Σ if the state x'_j is stably reachable from the state x'_i in Σ' . The isomorphism $\Psi : \Sigma_{c|s} \rightarrow \Sigma'$ between the machines is slightly different than before. The isomorphism $\Psi = \{\Psi_1, \Psi_2, \Psi_3\}$ consists of $\Psi_2 : A \rightarrow A$ and $\Psi_3 : Y \rightarrow Y$, two identity maps as before. The state map $\Psi_1 : (X \times \alpha) \rightarrow X'$ relabels the states of the stable state closed loop system $\Sigma_{c|s}$ to correspond with the states of Σ' by $\Psi_1((x_i, \alpha)) = x'_i$ for all $x_i \in X$ and $x'_i \in X'$, here again α is a state of the controller. The state map $\Psi_1 : (X \times \alpha) \rightarrow X'$ can be split into two separate maps as follows: $\Psi_1 = \Psi' \Psi_\alpha$, where $\Psi' : X \rightarrow X'$ by $\Psi'(x_i) = x'_i$ and $\Psi_\alpha : X \times \alpha \rightarrow X$ by $\Psi_\alpha((x_i, \alpha)) = x_i$.

(7.2.3.) PROPOSITION. Let $M = \{\Sigma_1, \Sigma_2, \dots, \Sigma_k\}$ be a critical race family of q machines induced by a machine $\Sigma = (A, Y, X, f, h)$. Let $\Sigma' = (A, Y, X', s', h)$ be a stable state machine and let $\#X = \#X' = n$. Pick any machine Σ_i from M , $i = 1, \dots, q$. If the skeleton matrix $S^{(n-1)}(\Sigma_i)$ satisfies $S^{(n-1)}(\Sigma_i) \geq S^{(n-1)}(\Sigma')$ then there exists a controller C_i such that the stable state closed loop system $\Sigma_{ic|s}$ is isomorphic to the machine Σ' .

(7.2.4) PROPOSITION. Let $M = \{\Sigma_1, \Sigma_2, \dots, \Sigma_q\}$ be a critical race family of machines induced by a machine $\Sigma = (A, Y, X, f, h)$ that has a single critical race. Let $\Sigma' = (A, Y, X', s', h)$ be a stable state machine and let $\#X = \#X' = n$.

If the skeleton matrix $\mathcal{H}(M)$ satisfies $\mathcal{H}(M) \geq S^{(n-1)}(\Sigma')$ then there is a controller C such that all members of $M_{c|s}$ are isomorphic to Σ' .

The proofs for sufficiency of these four propositions are similar to those given in the proofs of Theorem (5.1.3), Lemma (5.2.2), Proposition (5.3.1), and Theorem (5.3.2). The proof for necessity is at this time not complete.

It can be shown that a controller can "add" states to the closed loop system of a trivial machine (one with zero states). The controller states can be observed at the output. For a nontrivial machine, the controllers constructed in this work do not create any new potentially stable states, but they can create new stable combinations. At this time, it is not clear whether some other controller can "add" states to a nontrivial machine, thus necessity for the propositions (7.2.1), (7.2.2), (7.2.3), and (7.2.4) has not been shown at this time.

There exist isomorphisms which not only relabel the states of a machine, but also change the ordering the states of the machine. We will now extend the results of this subsection to a state mapping that reorders and relabels the states of a machine. First we define a needed notion.

Define the term *similar* for two skeleton matrices S and S' as follows: S is *similar* to S' if there exists a similarity transform T such that $TST^{-1} = S'$. We are interested in transformations which correspond to permutations of the states of a machine. A permutation of the states of a machine Σ induces a similarity transform T on the one-step matrix of stable transitions $R(\Sigma)$ and the one-step skeleton matrix $S(\Sigma)$. A permutation of the states of a machine Σ also induces the same similarity transformation T on the m -step matrix of stable transitions $R^m(\Sigma)$, the m -

step skeleton matrix $S^m(\Sigma)$, the matrix of stable transitions, and the skeleton matrix. The matrix T induced by a permutation of the states of a machine has a single one in each row and column.

(7.2.5) PROPOSITION. Let $\Sigma = (A, Y, X, f, h)$ be a machine. An isomorphism Ψ acting on Σ induces a similarity transform T on the one-step matrix of stable transitions $R(\Sigma)$ and the one-step skeleton matrix $S(\Sigma)$ of the machine Σ .

PROOF. Denote the states of Σ as $X = \{x_0, x_1, \dots, x_{n-1}\}$. Let $\Psi = \{\Psi_1, \Psi_2, \Psi_3\}$ be an isomorphism which maps Σ to a machine $\Sigma' = (A, Y, X', f, h')$. Denote the states of Σ' as $X' = \{x'_0, x'_1, \dots, x'_{n-1}\}$. The state map $\Psi_1 : X \rightarrow X'$ can reorder the states of X and relabel them as the states of X' by mapping each state $x_i \in X$ to a corresponding state $x'_j \in X'$, $\Psi_1(x_i) = x'_j$. The state map $\Psi_1 : X \rightarrow X'$ can be split into two separate maps as follows: $\Psi_1 = \Psi' \Psi_T$ where $\Psi' : X \rightarrow X'$ by $\Psi'(x_i) = x'_i$ and $\Psi_T : X \rightarrow X$ by $\Psi_T(x_i) = x_j$. Looking at an isomorphism acting on a stable state closed loop system, the state map $\Psi_1 : X \times \alpha \rightarrow X'$ can be split into three separate maps as follows: $\Psi_1 = \Psi' \Psi_T \Psi_\alpha$ where $\Psi' : X \rightarrow X'$ by $\Psi'(x_i) = x'_i$, $\Psi_T : X \rightarrow X$ by $\Psi_T(x_i) = x_j$, and $\Psi_\alpha : X \times \alpha \rightarrow X$ by $\Psi_\alpha((x_i, \alpha)) = x_i$. In qualitative terms the map Ψ_α removes the controller state, the map Ψ_T reorders the states, and the map Ψ' relabels the states. The similarity transform T induced by the isomorphism Ψ is induced by the map Ψ_T .

Recall the one-step matrix of stable transitions and the one-step skeleton matrix are constructed based on the order of the states of Σ . Switching the order of two of the states of Σ , x_i and x_j , corresponds to switching row i with row j and column i with column j in both the one-step matrix of stable transitions and the one-step skeleton matrix of the machine

Σ . Row and column switches of a matrix can be performed by a similarity transform, thus the isomorphism induces a similarity transform on the one-step matrix of stable transitions and the one-step skeleton matrix. ♦

Given an isomorphism Ψ we can construct the corresponding similarity transform T , and given a similarity transform T we can construct the corresponding isomorphism Ψ .

Let $\Sigma = (A, Y, X, f, h)$ be a machine, and let $\Sigma' = (A, Y, X', f, h')$ be another machine. Denote the states of Σ by $X = \{x_0, x_1, \dots, x_{n-1}\}$ and denote the states of Σ' by $X' = \{x_0', x_1', \dots, x_{n-1}'\}$. Let $\Psi : \Sigma \rightarrow \Sigma'$ be an isomorphism such that $\Psi_1 : X \rightarrow X'$. Define the vectors $\overline{x}$ and $\overline{x'}$ by

$$\overline{x} = \begin{pmatrix} x_0 \\ x_1 \\ . \\ . \\ x_{n-1} \end{pmatrix} \quad \overline{x'} = \begin{pmatrix} x_0' \\ x_1' \\ . \\ . \\ x_{n-1}' \end{pmatrix}$$

We construct the matrix T from an isomorphism $\Psi : \Sigma_s \rightarrow \Sigma'$ as follows. Start with the matrix T equal to the identity matrix, $T = I$. Then, perform row switches on T that correspond to the desired state reorderings. This will give $T\overline{x} = \overline{x'}$ to a relabeling of the states. Consider Ψ which takes the state x_i of Σ to the state x_j' of Σ' , i.e. $\Psi_1(x_i) = x_j'$. For this case $\Psi_1(x_i) = x_j'$, the matrix T is the identity matrix with rows i and j switched. By switching all corresponding rows to the states that are switched we can form the matrix T for any reordering of the states of Σ . Another way of looking at the matrix T is, if $\Psi_1(x_i) = x_j'$ then the i th row of T , T_{ij} , has a one in its j th column and zeros elsewhere in the row.

Given a similarity transform T , we can construct an isomorphism $\Psi : \Sigma \rightarrow \Sigma'$ where $\Psi_1 : X \rightarrow X'$ as follows. Set $\Psi_1(x_i) = x_j'$ if the (i,j) entry of the matrix T , T_{ij} , is a one. Continue this for all non zero entries and the full Ψ_1 will have been constructed.

The operation of the similarity transform T on matrices of stable transitions and skeleton matrices is performed as follows. Let T be a matrix of ones and zeros, and let R be a matrix over $S(A) \cup N$. Define the operation mult for $r \in S(A) \cup N$ and the integers 1 and 0 by

$$\begin{aligned} \text{mult}(1,r) &= r & \text{mult}(r,1) &= r \\ \text{mult}(0,r) &= N & \text{mult}(r,0) &= N \end{aligned}$$

Now consider two $n \times n$ matrices R and T described above. The result V of TR is defined by

$$V_{ij} := \bigcup_{k=1}^n \text{mult}(T_{ik}, R_{kj}), \quad i, j = 1, \dots, n.$$

And the result W of RT is defined by

$$W_{ij} := \bigcup_{k=1}^n \text{mult}(R_{ik}, T_{kj}), \quad i, j = 1, \dots, n.$$

And TRT^{-1} can be found by $((TR)T^{-1})$

The similarity transform is performed as usual for the skeleton matrix.

(7.2.6) EXAMPLE. Let $\Sigma = (A, Y, X, f, h)$ be a machine. Denote the states of Σ by $X = \{x_0, x_1, x_2\}$, and denote the input set of Σ by $A = \{a, b, c\}$. Define the vector $\overline{x}$ by

$$\overline{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix}$$

Let the skeleton matrix and the matrix of stable transitions of Σ be

$$S^{(2)}(\Sigma) = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \quad R^{(2)}(\Sigma) = \begin{pmatrix} \{a,c\} & \{b\} & N \\ \{a\} & \{b\} & \{c\} \\ \{a\} & N & \{b,c\} \end{pmatrix}$$

Let Ψ be an isomorphism operating on Σ such that $\Psi_1 : X \rightarrow X$ given by

$$\Psi_1(x_0) = x_0 \quad \Psi_1(x_1) = x_2 \quad \Psi_1(x_2) = x_1$$

The similarity transform T induced by Ψ is

$$T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} = T^{-1}$$

And $T \overline{x}$ is given by

$$T \overline{x} = \begin{pmatrix} x_0 \\ x_2 \\ x_1 \end{pmatrix}$$

$TS^{(2)}(\Sigma)T^{-1}$ is then

$$TS^{(2)}(\Sigma)T^{-1} = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

and $TR^{(2)}(\Sigma)T^{-1}$ is

$$TR^{(2)}(\Sigma)T^{-1} = \begin{pmatrix} \{a,c\} & N & \{b\} \\ \{a\} & \{b,c\} & N \\ \{a\} & \{c\} & \{b\} \end{pmatrix} \blacklozenge$$

We can now restate Proposition (7.2.2).

(7.2.7) PROPOSITION. Let $\Sigma|_S = (A, Y, X, s, h)$ be the stable state machine induced by a machine $\Sigma = (A, Y, X, f, h)$, and let $\Sigma' = (A, Y, X', s', h')$ be a stable state machine. Let $\#X = \#X' = n$. If there is a matrix S similar to the skeleton matrix $S^{(n-1)}(\Sigma)$ of the machine Σ such that $S \geq S^{(n-1)}(\Sigma')$, then there is a controller C such that the stable state closed loop system $\Sigma_{c|S}$ is isomorphic to the machine Σ' .

7.3 Matching Smaller Machines

Recall that the model matching problem involves finding a controller such that the closed loop system of (1.1) is equivalent to a model which is the same size as the machine in terms of the number of states, the size of the input sets, and the size of the output sets. We may also want to match a model smaller than our machine. This leads to the definition of a submachine.

Let $\Sigma = (A, Y, X, f, h)$ be a machine and let $X' \subset X$ be any subset of X containing x_0 the initial state. Define f_r for all valid pairs $(x', u) \in X' \times A$ as follows

$$f_r(x', u) = \begin{cases} f(x', u) & \text{if } f(x', u) \in X' \\ \emptyset & \text{otherwise} \end{cases}$$

Then define the output function $h_r : X' \times A \rightarrow Y$ for all valid pairs $(x', u) \in X' \times A$ by setting

$$h_r(x', u) = \begin{cases} h(x', u) & \text{if } h(x', u) \neq \emptyset \\ \emptyset & \text{otherwise} \end{cases}$$

There results a sequential machine $\Sigma|X' = (A, Y, X', f_r, h_r)$ called the restriction of Σ to X' . The surjective partial function $\Psi_r : X \rightarrow X'$, defined by $\Psi_r[x] = x$ if $x \in X'$ and $\Psi_r[x] = \emptyset$ otherwise, is then a state mapping called the *restriction mapping*. This state mapping is proper whenever $f(X', A) \subset X'$.

Let $\Sigma = (A, Y, X, f, h)$ and $\Sigma' = (A', Y', X', f', h')$ be two machines. The machine Σ' is a *submachine* of Σ whenever the following holds:

- (i) $X' \subset X$, $A' \subset A$, $Y' \subset Y$
- (ii) $f(x, u) = f'(x, u)$ restricted to $x \in X'$, $u \in A'$ and
- (iii) $h(x, u) = h'(x, u)$ restricted to $x \in X'$, $u \in A'$.

Let $\Sigma = (A, Y, X, f, h)$ and $\Sigma' = (A', Y', X', f', h')$ be two machines. The machine Σ' is a *generalized submachine* of the machine Σ if there exists a machine Σ_1 isomorphic to Σ' , such that Σ_1 is a submachine of Σ .

We can now state the following result.

(7.3.1) PROPOSITION. Let $\Sigma|_s = (A, Y, X, s, h)$ and $\Sigma'_s = (A, Y, X', s', h)$ be two stable state machines induced by the machines Σ and Σ' respectively. Let $X' \subset X$, $|X| = n$, and $|X'| = m$. If there is a matrix S similar to $S^{(n-1)}(\Sigma)$ such that the left upper $m \times m$ sub matrix S_m of S satisfies $S_m \geq S^{(m-1)}(\Sigma')$ then there exists a controller C such that a m -dimensional submachine of the stable state closed loop system $\Sigma_{c|s}$ is equivalent to the machine Σ'_s .

For $\Sigma_{c|s}$ to be isomorphic to Σ'_s , we need a bijective state map Ψ_1 which satisfies $\Psi_1 : (X \times \alpha) \rightarrow X'$. Denote the n states of X by $X = \{x_0, x_1, \dots, x_{n-1}\}$ and a state of our controller as α . The set of states for the stable state closed loop system is then $(X \times \alpha) = \{(x_0, \alpha), (x_1, \alpha), \dots, (x_{n-1}, \alpha)\}$. The

states of our controller do not add to the cardinality of the state set of the closed loop system, $|(X \times \alpha)| = |X|$. Thus the closed loop system $\Sigma_{c|s}$ has the same number of states as the machine $\Sigma_{|s}$. This implies that we require $n \geq m$ in the previous proposition.

This cardinality requirement is also true for input and output sets of $\Sigma_{|s}$ and $\Sigma_{|s}'$ if they differ. Consider the two machines $\Sigma_{|s} = (A, Y, X, s, h)$ and $\Sigma_{|s}' = (A', Y', X', s', h')$. If $|A| \geq |A'|$ the input characters used to match the model are restricted to the subset of A in the submachine of $\Sigma_{c|s}$ that is mapped into A' by the isomorphism. If $|Y| \geq |Y'|$ the output characters used to match the model are restricted to the subset of Y in the submachine of $\Sigma_{c|s}$ that is mapped into Y' by the isomorphism.

7.4 General Families of Machines

There is no guarantee that a controller would not need to provide two different outputs for the same state and reference input by only looking at the skeleton matrix for a general family of machines. Recall the skeleton matrix identifies all possible stable transitions between potentially stable states of a machine, but not the inputs required or the paths taken for the stable transitions.

There may not be any similarity in the transition functions of the members of a family of machines. Thus, even though a particular transition may be possible in all members of the family, the paths or the input lists required for the transition may have no similarity. In certain cases, a controller will not be able to recognize which machine it is controlling, and thus not be able to provide the correct values to the machine it is controlling.

The controller designed in Theorem (5.3.2) recognizes which member of the family it is controlling, after it encounters a critical race. Thus it is able to provide the correct values to the machine it is controlling.

Consider two machines Σ_1 and Σ_2 with the same input set A and the same state set X of a family M . Let both machines have a stable transition from the potentially stable state x to the potentially stable state x' . Let the machine Σ_1 have the stable transition $s_1(x,u) = x'$ and the machine Σ_2 have the stable transition $s_2(x,u') = x'$, where $x, x' \in X$, $u, u' \in A$, and $u \neq u'$. The skeleton matrix of M will have a one in the entry corresponding to the states x and x' , but a controller would need to provide two different outputs for the same state and reference input in this case to drive both machines of M from x to x' . This is not possible for our controller.

Controller existence can be determined for a general family of machines, but it requires looking at more than the skeleton matrices. A method of existence using the matrix of stable transitions should be able to solve this problem.

7.5 Strictly Causal Controllers

In this work we have constructed causal controllers for strictly causal machines. This ensures that our closed loop system of (1.1) is well posed. We could, however, construct a strictly causal controller and achieve the same results as with the causal controllers constructed previously in this work. There is a drawback, however. A strictly causal controller constructed by our algorithm requires more controller states than the causal controller for the same model matching problem. This does not

mean that there is no strictly causal controller of similar or smaller size to our causal controller, but the algorithm presented here generates a smaller causal controller than strictly causal controller for the same model matching problem.

Consider the controller of Theorem (5.1.3) for a machine $\Sigma = (A, Y, X, f, h)$, and suppose we would like to alter k transitions of Σ . Our strictly causal controller requires $\#A$ states in place of the single initial state α_0 of the controller C in (5.1.3) and $|k|$ states in place of the single state α_1 of the controller C of (5.1.3).

Our strictly causal controller requires the extra states because its output is dependent only upon the current controller state. Thus, for the controller to be able to output all possible reference inputs, the controller needs one state for each of the inputs of Σ . Similarly, for the controller to output the correct value for the stable combination of a pair to alter, the controller needs one state for each of these input values. The rest of the strictly causal controller is constructed similarly to the causal controller, with the exception of the output function which is defined only at each state of the controller.

7.6 Other Work

In the present work, a specific output function was investigated, i.e. the output of a machine is isomorphic to the current state of the machine. Our controller requires the current state of the machine to generate the appropriate output lists. An observer is needed to provide the controller with the present state of the machine if the output of the machine is not the current state. Not only would the comparison of the transition maps of two

machines be needed for the model matching problem, but the output maps of the two machines would also need to be compared. This restricts the machine models that can be matched with a corrective controller, and the existence of such a corrective controller will also depend on the observability of the states of the machine being controlled.

The corrective controller constructed in this work ensures that the closed loop system of (1.1) operates in fundamental mode. This requires the restriction of the machines controlled to those without any unstable cycles. A controller which operates significantly faster than the machine it is controlling may allow the correction of such problems. Control could then be applied during the unstable transitions of the machine without creating any new races. We would allow a violation of fundamental mode because the controller could change the inputs to the machine significantly faster than the machine could change states. In this work, input changes were only allowed during stable combinations. This requires some modification of this work, the matrix of transitions and skeleton matrices need to be modified to reflect all the transitions of the machine, not just the stable transitions.

LIST OF REFERENCES

- [1] Arbib, M., *Theories of Abstract Automata*, Prentice Hall Inc., Englewood Cliffs, NJ, 1969.
- [2] Armstrong, D., Friedman A., and Menon, P., Realization of Asynchronous Sequential Circuits without Inserted Delay Elements, *IEEE Transactions on Computers*, C-17, no. 2, February, 1968.
- [3] Bredeson, J., and Hulina, P., Generation of a Clock Pulse for Asynchronous Sequential Machines to Eliminate Critical Races, *IEEE Transactions on Computers*, C-20, pp. 225-226, February, 1971.
- [4] Brzozowski, J.A., and Seger, C., A Characterization of Ternary Simulation of Gate Networks, *IEEE Transactions on Computers*, C-36, no. 11, pp. 1318-1327, November, 1987.
- [5] Brzozowski, J.A., and Seger, C.-J., A Unified Framework for Race Analysis of Asynchronous Networks, *Journal of the Association for Computing Machinery*, vol. 36, no. 1, pp. 20-45, January, 1989.
- [6] Brzozowski, J.A., and Yoeli, M., On a Ternary Model of Gate Networks, *IEEE Transactions on Computers*, C-28, pp. 178-183, March, 1979.
- [7] Chiang, J., and Radhakrishnan, D., Hazard-free Design of Mixed Operating Mode Asynchronous Sequential Circuits, *International Journal of Electronics*, vol. 68, no. 1, pp. 23 - 37, January, 1990.
- [8] Chu, T., Mani, N., and Leung, C., An Efficient Critical Race-Free State Assignment Technique for Asynchronous Finite State Machines, *Proceedings of the 30th ACM/IEEE Design Automation Conference*, Dallas, TX, pp. 2-5, 1993.
- [9] Datta, P.K., Bandyopadhyay, S.K., and Choudhury, A.K., A Graph Theoretic Approach for State Assignment of Asynchronous Sequential Machines, *International Journal of Electronics*, vol. 65, no. 6, pp. 1067 - 1075, December, 1988.
- [10] Eichelberger, E.B., Hazard Detection in Combinational and Sequential Switching Circuits, *IBM Journal of Research and Development*, vol 9, no. 2, pp. 90-99, 1965.
- [11] Eilenberg, S., *Automata, Languages, and Machines*, volume A, Academic Press, New York, 1974.

- [12] Evans, J.B., *Structures of Discrete Event Simulation*, Halsted Press, New York, 1988.
- [13] Fisher, P., and Wu, S., Race-Free State Assignments for Synthesizing Large-Scale Asynchronous Sequential Logic Circuits, *IEEE Transactions on Computers*, vol. 42, no. 9, pp. 1025-1034, September, 1993.
- [14] Hammer, J., On Corrective Control of Sequential Machines, *International Journal of Control* (in press).
- [15] Hammer, J., On Some Control Problems in Molecular Biology, *Proceedings of the IEEE Conference on Decision and Control*, New Orleans, LA, pp. 3747-3752, December, 1994.
- [16] Hammer, J., On the Control of Incompletely Described Sequential Machines, *International Journal of Control* (in press).
- [17] Hammer, J., On the Modeling and Control of Biological Signaling Chains, *Proceedings of the IEEE Conference on Decision and Control*, Lake Buena Vista, Fl, pp. 4098-4103, December, 1995.
- [18] Hammer, J., On Biological Signaling Chains and their Control, submitted for publication, 1995.
- [19] Hazeltine, B., Encoding of Asynchronous Sequential Circuits, *IEEE Transactions on Electronic Computers*, EC-14, pp. 727-729, 1965.
- [20] Hlavicka, J., Essential Hazard Correction without the use of Delay Elements, *IEEE Transactions on Computers*, C-19, no. 3, pp. 232-238, March, 1970.
- [21] Huffman, D.A., The Synthesis of Sequential Switching Circuits, *J. Franklin Inst.*, vol. 257, pp. 161-190, March 1954, pp. 275-303, April 1954.
- [22] Huffman, D. A., The Design and Use of Hazard-Free Switching Networks, *Journal of the Association of Computing Machinery*, vol. 4, no. 1, pp. 47-62, January, 1957.
- [23] Kalman, R.E., Falb, P.L., and Arbib, M.A., *Topics in Mathematical System Theory*, McGraw-Hill Book Company, New York, 1969.
- [24] Kohavi, Z, *Switching and Finite Automata Theory*, McGraw-Hill Book Company, New York, 1970.

- [25] Lavagno, L., Keutzer, K., and Sangiovanni-Vincentelli, A., Algorithms for Synthesis of Hazard-Free Asynchronous Circuits, *Proceedings of the 28th ACM/IEEE Design Automation Conference*, San Francisco, CA, pp. 302-308, 1991.
- [26] Lin, F., and Wonham, W.M., Decentralized Control and Coordination of Discrete Event Systems with Partial Observation, *IEEE Transactions on Automatic Control*, vol. 35, no. 12, pp. 1330 - 1337, December, 1990.
- [27] Liu, C., A State Variable Assignment Procedure for Asynchronous Sequential Switching Circuits, *Journal of the Association of Computing Machinery*, vol. 10, pp. 209-216, April, 1963.
- [28] MacLane, S., and Birkhoff, G., *Algebra*, The Macmillan Company, New York, 1967.
- [29] Maki, G., and Tracey, J., A State Assignment Procedure for Asynchronous Sequential Circuits, *IEEE Transactions on Computers*, C-20, pp. 666-668, June, 1971.
- [30] Mago, G., Realization of Methods for Asynchronous Sequential Circuits, *IEEE Transactions on Computers*, C-20, pp. 290-298, March, 1971.
- [31] McCluskey, E.J., Fundamental Mode and Pulse Mode Sequential Circuits, Proc. IFIP Congress 1962, *Intern. Conf. Information Processing*, Munich, Aug. 27-Sept. 1, 1962, pp. 725-730, Cicely M. Popplewell (ed.), North Holland Publishing Company, Amsterdam, 1963.
- [32] McCluskey, E.J., *Introduction to the Theory of Switching Circuits*, McGraw-Hill Book Company, New York, 1965.
- [33] Mealy, G.H., A Method for Synthesizing Sequential Circuits, *Bell System Tech. J.*, vol. 34, pp. 1045-1079, September, 1955.
- [34] Moore, E.F., Gedanken-experiments on Sequential Machines, *Automata Studies, Annals of Mathematical Studies*, no. 34, Princeton University Press, Princeton, N.J., 1956.
- [35] Oikonomou, K.N., Abstractions of Finite-State Machines and Immediately-Detectable Output Faults, *IEEE Transactions on Computers*, vol. 41, no. 3, pp. 325-338, March, 1992.
- [36] Ozveren, C.M., and Willsky, A.S., Observability of Discrete Event Dynamic Systems, *IEEE Transactions on Automatic Control*, vol. 35, no. 7, pp. 797 - 806, July, 1990.

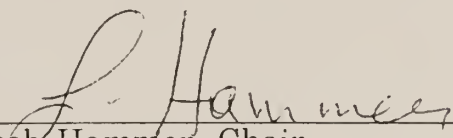
- [37] Ozveren, C.M., Willsky, A.S., and Antsaklis, P.J., Stability and Stabilizability of Discrete Event Dynamic Systems, *Journal of the Association of Computing Machinery*, vol. 38, no. 3, pp. 730 - 752, 1991.
- [38] Piguet, C., Logic Synthesis of Race-Free Asynchronous CMOS Circuits, *IEEE Journal of Solid-State Circuits*, vol. 26, no. 3, pp. 371-380, March, 1991.
- [39] Ramadge, P.J.G., and Wonham, W.M., Supervisory Control of a Class of Discrete Event Processes, *SIAM Journal of Control and Optimization*, vol. 25, no. 1, pp. 206 - 230, January, 1987.
- [40] Ramadge, P.J., and Wonham, W.M., The Control of Discrete Event Systems, *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81 - 98, January, 1989.
- [41] Ramadge, P.J., and Wonham, W.M., Modular Feedback Logic for Discrete Event Systems, *SIAM Journal of Control and Optimization*, vol. 25, pp. 1202 - 1218, September, 1987.
- [42] Tracey, J.H., Internal State Assignments for Asynchronous Sequential Machines, *IEEE Transactions on Electronic Computers*, EC-15, pp. 551-560, August, 1966.
- [43] Turing, A, On Computable Numbers, with an Application to the Entscheidungs-Problem, *Proceedings of the London Mathematical Society*, Ser. 2-42, pp. 230-265 with a correction, Ibid, Ser. 2-43, pp. 544-546, 1936-1937.
- [44] Unger, S.H., Hazards and Delays in Asynchronous Sequential Switching Circuits, *IRE Transactions on Circuit Theory*, vol. CT-6, no. 1, pp. 12-25, March, 1959.
- [45] Whitaker, S., and Maki, G., Self Synchronized Asynchronous Sequential Pass Transistor Circuits, *IEEE Transactions on Computers*, vol. 41, no. 10, pp. 1344-1348, October, 1992.
- [46] Wonham, W.M., and Ramadge, P.J.G., On the Supremal Controllable Sublanguage of a Given Language, *SIAM Journal of Control and Optimization*, vol. 25, pp. 637 - 659, May, 1987.
- [47] Yoeli, M., and Rinon, S., Application of Ternary Algebra to the Study of Static Hazards, *Journal of the Association of Computing Machinery*, vol. 11, no. 1, pp. 84-97, January, 1964.

BIOGRAPHICAL SKETCH

Thomas E. Murphy was born in Elmira, New York, and grew up in Painted Post, New York, and Corning, New York. He graduated from Corning Painted Post West High School in 1984.

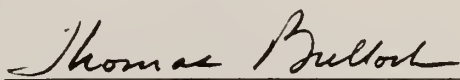
He attended the University of Notre Dame, Notre Dame, Indiana, where he received a B.S. in electrical engineering in 1988. He received his M.E. in electrical engineering from the University of Florida, Gainesville, Florida, in 1992. Since then he has been at the University of Florida.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



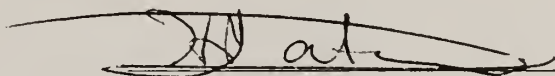
Jacob Hammer, Chair
Professor of Electrical and
Computer Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



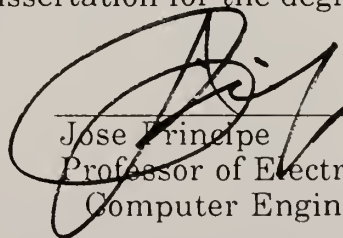
Thomas Bullock
Professor of Electrical and
Computer Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Haniph Latchman
Associate Professor of Electrical
and Computer Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.



Jose Principe
Professor of Electrical and
Computer Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

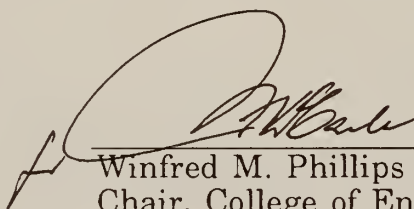


John Schueller

Associate Professor of Mechanical
Engineering

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

August, 1996



Winfred M. Phillips

Chair, College of Engineering

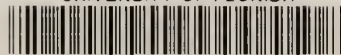
Karen A. Holbrook

Dean, Graduate School

LD
1780
1996
.M978

SCIENCE
LIBRARY

UNIVERSITY OF FLORIDA



3 1262 08554 4269

